

Regular path queries on graphs with data: A rigid approach*

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
wuzl@ios.ac.cn

ABSTRACT

Regular path queries (RPQ) is a classical navigational query formalism for graph databases to specify constraints on labeled paths. Recently, RPQs have been extended by Libkin and Vrgoč to incorporate data value comparisons among different nodes on paths, called regular path queries with data (RDPQ). It has been shown that the evaluation problem of RDPQs is PSPACE-complete and NLOGSPACE-complete in data complexity. On the other hand, the containment problem of RDPQs is in general undecidable. In this paper, we propose a novel approach to extend regular path queries with data value comparisons, called rigid regular path queries with data (RRDPQ). The main ingredient of this approach is an automata model called nondeterministic rigid register automata (NRRRA), in which the data value comparisons are *rigid*, in the sense that if the data value in the current position x is compared to a data value in some other position y , then by only using the labels (but not data values), the position y can be uniquely determined from x . We show that NRRAs are robust in the sense that nondeterministic, deterministic and two-way variant of NRRAs, as well as an extension of regular expressions, are all of the same expressivity. We then argue that the expressive power of RDPQs are reasonable by demonstrating that for every graph database, there is a localized transformation of the graph database so that every RDPQ in the original graph database can be turned into an equivalent RRDPQ over the transformed one. Finally, we investigate the computational properties of RRDPQs and conjunctive RRDPQs (CRRDPQ). In particular, we show that the containment of CRRDPQs (and RRDPQs) can be decided in 2EXPSpace.

1. INTRODUCTION

Graph data management is a classical research field in database community and has achieved a recent resurgence, with the momentums from new application domains, such as online social networks, bioinformatics, and semantic web. Various query languages have been proposed for graph databases (see [2, 21, 3] for surveys). Among them, regular path queries (RPQ) are

basic query formalisms to specify path constraints in graph databases.

Graph databases are usually modelled as edge-labeled graphs. A RPQ looks for a pair of nodes connected by a path whose sequence of labels belongs to a regular language ([9]). For the convenience of specifications, RPQs can be extended in a natural way, called RPQs with inverse word symbols (2RPQ), to allow traversing edges in both directions. Since the availability of inverse symbols much eases the specifications, we will focus on path queries with inverse symbols through this paper.

Extensions of RPQs to specify the relationships among multiple paths have been investigated intensively, e.g. conjunctive RPQs (CRPQ) which specify the existence of several paths on the whole ([12, 8, 11]), nested regular expressions where multiple RPQs are organized into a tree structure ([19, 6]), extended CRPQs where regular or rational relations over paths are allowed ([5, 4]).

RPQs have also been extended in another way, called regular data path queries (RDPQ), to incorporate data value comparisons between two nodes in a path ([18]). RDPQs are interpreted over data graphs, which extend graph databases by assigning a data value to every node. A RDPQ looks for a pair of nodes connected by a path whose sequence of data values and labels is accepted by a nondeterministic register automata (NRA). NRA is an extension of finite state automata, where a fixed number of registers are used to store the data values. Similarly to 2RPQs and CRPQs, RDPQs with inverse (2RDPQ), conjunctive RDPQs (CRDPQ), or conjunctive 2RDPQs (C2RDPQ), can also be defined.

Evaluation and containment are two basic problems for database query languages. These two problems have been investigated extensively for RPQs and 2RPQs, CRPQs and C2RPQs (see [3] for a survey). For RDPQs, the evaluation problem is PSPACE-complete, and NLOGSPACE-complete in data complexity. On the other hand, the containment problem of RDPQs is undecidable, as a result of the undecidability of the inclusion problem of NRAs ([18]). Since the containment and equivalence problem are essential for the optimization of queries, this undecidability result of RDPQs seems

*Supported by the National Natural Science Foundation of China under Grant No. 61100062.

to undermine the validity of RDPQs as a fundamental formalism of path queries that combines the labelling and data constraints.

Our goal in this paper is to propose an alternative extension of 2RPQs with data value comparisons, called rigid regular data path queries with inverse (2RRDPQ), which, we believe, achieves a good balance between the expressive power and the computational properties (decidability and complexity).

2RRDPQs are based on an automaton model also proposed in this paper, called nondeterministic rigid register automata (NRRRA), where the data value comparisons are “rigid” in the sense that if the data value in the current position x is compared to a data value in some other position y , then by only using the labels (but not data values), the position y can be uniquely determined from x . With the rigidity constraint, we are able to show that NRRAs enjoy nice properties as finite state automata, that is, NRRAs can be determined, they are closed under all Boolean operations, the two-way variant of NRRAs is expressively equivalent to (one-way) NRRAs, and they are expressively equivalent with a natural extension of regular expressions. In addition, while the expressive power of NRRAs and NRAs are incomparable, we demonstrate that the expressive power of NRRAs can be captured by an extension of NRAs with nondeterministic guessing.

To justify the expressibility of 2RRDPQs, we show that although the expressive power of 2RRDPQs and 2RPQs are incomparable, every 2RPQ can in fact be turned into a 2RRDPQ if a localized transformation is applied to graph databases. By “localized transformation”, we mean that the transformation is obtained by adding for each node v a new node n_v which is only connected to v and the global topology of the original graph is preserved (see Section 4).

We then investigate the computational properties of 2RRDPQs. We show that 2RRDPQs can be evaluated over data graphs with the same (data and combined) complexity as RDPQs. In addition, we consider conjunctive 2RRDPQs (C2RRDPQ) and show that the containment problem of C2RRDPQs can be decided in 2EXSPACE. From this, we deduce that the containment problem of 2RRDPQs can be decided in 2EXSPACE as well. The 2EXSPACE result is proved by a nontrivial extension of the proof for the EXSPACE-completeness result of C2RPQs in [8], and is the most technical part of this paper.

Related work. The idea of rigidity is inspired by event clock automata from the verification community ([1]), where for every event a , a clock x_a is used to record the time that has been elapsed from the last occurrence of a , and a clock y_a is used to predict the time that will elapse until the next occurrence of a . Although in spirit similar to event clock automata, NRRAs are defined to

allow much more complicated data value comparisons. For instance, in NRRAs, the current data value can be compared to the data value in the position corresponding to the last occurrence of the word symbol a before the next occurrence of the symbol b . This capability of data value comparisons is essential for the proof of the 2EXSPACE result of the containment problem of C2RRDPQs (see Section 5). NRAs were introduced in [13]. A restriction of NRAs, window memory automata, has been proposed in [7], where only local data value comparisons are allowed. NRRAs strictly extend window memory automata, since non-local data value comparisons are allowed. Various query formalisms have been proposed for data graphs to combine the topology and data constraints, e.g. XPath with data comparisons [15], TriAL for RDF [16]. Although the containment problem of data path queries is in general undecidable, it has been examined in detail for various fragments with positive data value comparisons ([15]).

Organization of this paper. Definitions are given in the next section. NRRAs and their variants are presented in Section 3. 2RRDPQs are investigated in Section 4. Section 5 deals with the C2RRDPQs.

2. DEFINITIONS

For a natural number k such that $k > 0$, let $[k]$ denote $\{1, \dots, k\}$ and $[-k]$ denote $\{-k, \dots, -1\}$.

Fix a finite alphabet Σ and an infinite set of data values \mathbb{D} . Let $\Sigma^\pm = \Sigma \cup \{a^- \mid a \in \Sigma\}$. For $a \in \Sigma^\pm$, we use a^- to denote the inverse of a . In particular, $(a^-)^- = a$ for $a \in \Sigma$.

A word w over the alphabet Σ is a finite sequence of elements from Σ . For a word w , $|w|$ is used to denote the length of w .

A *data path* α over Σ is a sequence $d_0 a_1 d_1 \dots a_n d_n$, where $d_0, \dots, d_n \in \mathbb{D}$, and $a_1, \dots, a_n \in \Sigma$. The data path of the minimum length is a single data value d . Given two data paths $\alpha_1 = d_0 a_1 d_1 \dots a_n d_n$ and $\alpha_2 = d_n a_{n+1} d_{n+1} \dots a_m d_m$, the *concatenation* of α_1 and α_2 , denoted by $\alpha_1 \cdot \alpha_2$, is defined as the following data path, $d_0 a_1 d_1 \dots a_n d_n a_{n+1} d_{n+1} \dots a_m d_m$. Note that $\alpha_1 \cdot \alpha_2$ is defined only if the last data value of α_1 is the same as the first data value of α_2 . The definition naturally extends to the concatenation of multiple data paths.

A *language* over Σ is a set of words over Σ and a *data language* over Σ is a set of data paths over Σ .

Let Σ and Γ be two finite alphabets. Then a *letter projection* prj from Σ to Γ is a *surjective* function from Σ to Γ . The letter projections of words, data paths, languages and data languages can be defined in a natural way. For a letter projection $prj : \Sigma \rightarrow \Gamma$ and $\gamma \in \Gamma$, we use $prj^{-1}(\gamma)$ to denote the set $\{a \in \Sigma \mid prj(a) = \gamma\}$. Note that $(prj^{-1}(\gamma))_{\gamma \in \Gamma}$ forms a partition of Σ . In addition, for $B \subseteq \Gamma$, let $prj^{-1}(B) = \bigcup_{\gamma \in B} prj^{-1}(\gamma)$.

A *graph database* \mathcal{G} is an edge-labeled graph (V, E) ,

where V is the set of nodes and $E \subseteq V \times \Sigma \times V$. For $e = (v, a, v') \in E$, let $\lambda(e)$ denote the label a . A *semipath* π in \mathcal{G} is a sequence $v_0 a_1 v_1 \dots v_{n-1} a_n v_n$ such that for every $i : 1 \leq i \leq n$, either $(v_{i-1}, a_i, v_i) \in E$ or $(v_i, a_i^-, v_{i-1}) \in E$. A *path* π in \mathcal{G} is a semipath $v_0 a_1 v_1 \dots v_{n-1} a_n v_n$ such that for every $i : 1 \leq i \leq n$, $(v_{i-1}, a_i, v_i) \in E$. Let $\lambda(\pi)$ denote the sequence of labels on a semipath π , that is, $a_1 \dots a_n$. A semipath π is *simple* if no nodes are repeated on π .

A *regular path query* (RPQ) over Σ is a tuple $\xi = (x, L, y)$, where L is a regular language over the alphabet Σ . The regular language L can be given by a finite state automaton or a regular expression. Given a graph database $\mathcal{G} = (V, E)$, the evaluation result of ξ over \mathcal{G} , denoted by $\xi(\mathcal{G})$, consists of the set of pairs (v, v') such that there is a path π from v to v' such that $\lambda(\pi) \in L$.

A *regular path query with inverse* (2RPQ) over Σ is a tuple $\xi = (x, L, y)$, where L is a regular language over the alphabet Σ^\pm . The semantics of 2RPQs are defined similarly to RPQs, with paths replaced by semipaths.

The evaluation problem for a RPQ or 2RPQ ξ , a graph database $\mathcal{G} = (V, E, \eta)$, a node pair (v, v') in \mathcal{G} , decide whether $(v, v') \in \xi(\mathcal{G})$.

The containment problem of a RPQ or 2RPQ is defined as follows: Let ξ_1, ξ_2 be two RPQs or 2RPQs. Then ξ_1 is contained in ξ_2 , denoted by $\xi_1 \subseteq \xi_2$, if for every graph database \mathcal{G} , $\xi_1(\mathcal{G}) \subseteq \xi_2(\mathcal{G})$.

A *conjunctive regular path query* (CRPQ) ξ over Σ is an expression of the form $\text{Ans}(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$,

where for every i , (y_{2i-1}, L_i, y_{2i}) is a RPQ over Σ , and \bar{z} is a tuple of variables from $\{y_1, \dots, y_{2l}\}$ (\bar{z} are called the *distinguished variables* of ξ). Note that in the above definition, y_i and y_j ($i \neq j$) may be the same variable.

Given a graph database $\mathcal{G} = (V, E)$, a CRPQ $\xi := \text{Ans}(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$, and $\nu : \{y_1, \dots, y_{2l}\} \rightarrow V$, we say $(\mathcal{G}, \nu) \models \xi$, if $(\nu(y_{2i-1}), \nu(y_{2i}))$ belongs to the evaluation result of (y_{2i-1}, L_i, y_{2i}) over \mathcal{G} for every $i : 1 \leq i \leq l$. The evaluation result of ξ over \mathcal{G} , denoted by $\xi(\mathcal{G})$, is the set of all tuples $\nu(\bar{z})$ for $\nu : \{y_1, \dots, y_{2l}\} \rightarrow V$ such that $(\mathcal{G}, \nu) \models \xi$. Similarly, C2RPQs can be defined, with RPQs replaced by 2RPQs.

The evaluation and containment problem of CRPQs or C2RPQs can be defined similarly to RPQs.

A *data graph* \mathcal{G} is a tuple (V, E, η) , where (V, E) is a graph database and $\eta : V \rightarrow \mathbb{D}$ assigns each node a data value. For a semipath $\pi = v_0 a_1 v_1 \dots v_{n-1} a_n v_n$ in (V, E) , the *data path* corresponding to π , denoted by $\eta(\pi)$, is $\eta(v_0) a_1 \eta(v_1) \dots \eta(v_{n-1}) a_n \eta(v_n)$.

Let k be a natural number. A *k-register data constraint* is defined by the following rules:

$$c := r_i \sim r_j \mid r_i \not\sim r_j \mid c \vee c \mid c \wedge c,$$

where $0 \leq i, j \leq k$, $i \neq j$, and r_0 is a special register

reserved for the current data value. Let \mathcal{C}_k denote the set of data constraints.

Let c be a data constraint, $\theta \in (\mathbb{D} \cup \{\perp\})^{[k]}$ be the current state of registers (where $\theta(i) = \perp$ denotes the fact that no data value is stored into the register r_i), and d be a data value, then the semantics of c is defined over (θ, d) as follows: If $c = r_i \sim r_j$, then $(\theta, d) \models c$ iff $\theta[0 \leftarrow d](i) \neq \perp$, $\theta[0 \leftarrow d](j) \neq \perp$, and $\theta[0 \leftarrow d](i) = \theta[0 \leftarrow d](j)$, where $\theta[0 \leftarrow d]$ is the function extending θ by assigning d to 0. The semantics of $c = r_i \not\sim r_j$ can be defined similarly. In addition, the semantics of $c = c_1 \vee c_2$ and $c = c_1 \wedge c_2$ are defined in a natural way.

Let k be a natural number. A *nondeterministic k-register data path automaton* (NRA, [18]) \mathcal{A} over Σ^\pm is a tuple (Q, k, δ, I, F) , where $Q = Q_w \cup Q_d$ such that Q_w and Q_d are two finite disjoint sets of word states and data states, k is the number of registers, $I \subseteq Q_d$ is the set of initial states, $F \subseteq Q_w$ is the set of final states, $\delta = \delta_w \cup \delta_d$ such that $\delta_w \subseteq Q_w \times \Sigma^\pm \times Q_d$ is the word transition relation and $\delta_d \subseteq Q_d \times \mathcal{C}_k \times Q_w \times 2^{[k]}$ is the data transition relation.

The intuition of the definition of NRAs is that since data paths alternate between data values and word symbols, when \mathcal{A} is in a data (resp. word) state, it is ready to read a data value (resp. a word symbol). Since data paths begin and end with data values, an initial state should be a state before reading a data value, so I is defined as a subset of Q_d , dually, a final state should be a state after reading a data value, so F is defined as a subset of Q_w .

Given a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$ and a NRA $\mathcal{A} = (Q, k, \delta, I, F)$, a *configuration* of \mathcal{A} on α is a tuple (q, j, θ) , where $q \in Q$, j is the current position (where $j = 0$ means the first position) of the symbol that \mathcal{A} reads, and $\theta \in (\mathbb{D} \cup \{\perp\})^{[k]}$ is the current state of the registers. An *initial configuration* of \mathcal{A} over α is $(q, 0, \theta_\perp)$, where $q \in I$, and $\theta_\perp(i) = \perp$ for every $i \in [k]$. Let $(q, j, \theta), (q', j+1, \theta')$ be two configurations (where $0 \leq j \leq 2n$). Then $(q', j+1, \theta')$ is said to be a *successor* of (q, j, θ) , denoted by $(q, j, \theta) \vdash_\alpha (q', j+1, \theta')$, if one of the following conditions holds.

- If the j -th symbol of α is a word symbol a , then $(q, a, q') \in \delta_w$, $\theta' = \theta$.
- If the j -th symbol of α is a data value d , then there are $c \in \mathcal{C}_k$, $X \subseteq [k]$ such that $(q, c, q', X) \in \delta_d$, $(\theta, d) \models c$, and θ' is obtained from θ by assigning d to every $i \in X$.

A data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$ is *accepted* by a NRA \mathcal{A} if there are $q \in I, q' \in F$ and a data assignment θ such that $(q, 0, \theta_\perp) \vdash_\alpha^* (q', 2n+1, \theta)$, where \vdash_α^* is the reflexive and transitive closure of \vdash_α . The set of data paths accepted by \mathcal{A} is denoted by $\mathcal{L}(\mathcal{A})$. In addition, for every $\theta, \theta' \in (\mathbb{D} \cup \{\perp\})^{[k]}$, we use $\mathcal{L}(\mathcal{A}, \theta, \theta')$ to denote the set of data paths $\alpha = d_0 a_1 d_1 \dots a_n d_n$ such that there

are $q \in I, q' \in F$ satisfying $(q, 0, \theta) \vdash_{\alpha}^* (q', 2n+1, \theta')$.

Let k be a natural number. *Regular expressions with k -memory* (REM, [18]) over Σ^{\pm} are defined by the following rules:

$$e := \varepsilon \mid \emptyset \mid a \mid e \cdot e \mid e \cup e \mid e^+ \mid \downarrow_X e \mid e[c],$$

where $a \in \Sigma^{\pm}$, $c \in \mathcal{C}_k$, and $X \subseteq [k]$.

The semantics of REMs is defined by a relation $\theta \vdash_{e, \alpha}$ θ' , where e is a REM, α is a data path, $\theta, \theta' \in (\mathbb{D} \cup \{\perp\})^{[k]}$. In the following, due to space constraints, we only present the semantics for the last two rules above, that is, $e = \downarrow_X e_1$ and $e = e_1[c]$, the semantics of the other rules are obvious and can be found in [18].

- If $e = \downarrow_X e_1$, then $\theta \vdash_{e, \alpha} \theta'$ if $\theta_{X=d} \vdash_{e_1, \alpha} \theta'$, where d is the first data value of α , and $\theta_{X=d}$ is obtained from θ by assigning d to all the registers in X .
- If $e = e_1[c]$, then $\theta \vdash_{e, \alpha} \theta'$ if $\theta \vdash_{e_1, \alpha} \theta'$ and $(\theta', d) \models c$, where d is the last data value of α .

A data path α is accepted by a REM e if there exists $\theta \in (\mathbb{D} \cup \{\perp\})^{[k]}$ such that $\theta_{\perp} \vdash_{e, \alpha} \theta$. The set of data paths accepted by a REM e is denoted by $\mathcal{L}(e)$. For every $\theta, \theta' \in (\mathbb{D} \cup \{\perp\})^{[k]}$, we use $\mathcal{L}(e, \theta, \theta')$ to denote the set of data paths α such that $\theta \vdash_{e, \alpha} \theta'$.

THEOREM 1 ([10, 18, 17]). *The following facts hold for NRAs and REMs.*

- NRAs and REMs are expressively equivalent.
- The nonemptiness problem of NRAs and REMs is PSPACE-complete.
- The universality and equivalence problem of NRAs and REMs are undecidable.

A *regular path query with data* (RDPQ) ξ over Σ is a tuple (x, L, y) , where L is a language of data paths defined by a NRA or a REM over the alphabet Σ . Given a data graph $\mathcal{G} = (V, E, \eta)$, the evaluation result of ξ over \mathcal{G} , denoted by $\xi(\mathcal{G})$, is the set of node pairs (v, v') in \mathcal{G} such that there is a path π from v to v' such that $\eta(\pi)$, the data path corresponding to π , belongs to L .

A *regular path query with inverse and data* (2RDPQ) ξ over Σ is a tuple (x, L, y) , where L is a language of data paths defined by a NRA or a REM over Σ^{\pm} . The semantics of 2RDPQ ξ over a data graph $\mathcal{G} = (V, E, \eta)$ is defined similarly to that of RDPQ, with paths replaced by semipaths.

Similarly to 2RPQs, regular path queries with inverse and data (2RDPQ) can be defined. Moreover, CRDPQs and C2RDPQs can be defined in the same way as CRPQs and C2RPQs. The evaluation and containment problem of RDPQs, 2RDPQs, CRDPQs, C2RDPQs can also be defined similarly.

THEOREM 2 ([18]). *The following results hold for RDPQs, 2RDPQs, CRDPQs and C2RDPQs.*

- The evaluation problem of RDPQs and 2RDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.
- The evaluation problem of CRDPQs and C2RDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.
- The containment problem of RDPQs, 2RDPQs, CRDPQs and C2RDPQs is undecidable.

3. RIGID REGISTER AUTOMATA AND ITS RELATIVES

In this section, we first define nondeterministic rigid register automata (NRRAs). Then we show the robustness of this model by proving that NRRAs can be determinized and their two-way as well as alternating variants are expressively equivalent to NRRAs. We also show that there is a natural extension of regular expressions equivalent to NRRAs.

3.1 Rigid data constraints

A *position term* t over the alphabet Σ^{\pm} is defined by the following rules,

$$t := \text{cur} \mid \text{suc}(t) \mid \text{pred}(t) \mid \text{suc}_A(t) \mid \text{pred}_A(t),$$

where A is a nonempty subset of Σ^{\pm} . Intuitively, the constant “cur” denotes the position of the current data value, “suc” and “pred” denote the position of the next and the previous data value, “suc_A” denotes the position of the data value *immediately after the next occurrence of a word symbol from A*, dually, “pred_A” denotes the position of the data value *immediately before the previous occurrence of a word symbol from A*.

Let $\mathcal{T}_p[\Sigma^{\pm}]$ denote the set of position terms over Σ^{\pm} .

For brevity, position terms of the form $\text{suc}_{\{a\}}(t)$ or $\text{pred}_{\{a\}}(t)$ (where $a \in \Sigma^{\pm}$) are written as $\text{suc}_a(t)$ or $\text{pred}_a(t)$. In addition, we use suc^i to denote the repetitions of suc for i times. Similarly, we use the abbreviations pred^i , suc_A^i , and pred_A^i .

The set of subterms of $t \in \mathcal{T}_p[\Sigma^{\pm}]$, denoted by $\text{sub}(t)$, are defined in a natural way, e.g. $\text{sub}(\text{suc}_A(t_1)) = \{\text{suc}_A(t_1)\} \cup \text{sub}(t_1)$. We use $t' \leq t$ to denote the fact that $t' \in \text{sub}(t)$, and $t' < t$ to denote the fact that $t' \leq t$ and $t \neq t'$. Suppose $t, t', t_1 \in \mathcal{T}_p[\Sigma^{\pm}]$ and $t' \leq t$, let $t[t' \setminus t_1]$ denote the position term obtained from t by replacing t' with t_1 .

The semantics of position terms are defined as follows: Give a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$ and a position $2i$ (where $0 \leq i \leq n$, $2i$ is the position for the data value d_i , and the first position is indexed by 0), the position represented by t over α and $2i$, denoted by $t_{\alpha}[2i]$, is defined as follows.

- $\text{cur}_{\alpha}[2i] = 2i$.
- If $i < n$, then $(\text{suc}(\text{cur}))_{\alpha}[2i] = 2(i+1)$. Otherwise, $(\text{suc}(\text{cur}))_{\alpha}[2i] = \perp$.

- If $i > 0$, then $(pred(cur))_\alpha[2i] = 2(i - 1)$. Otherwise, $(pred(cur))_\alpha[2i] = \perp$.

- If $(t_1)_\alpha[2i] \neq \perp$, then

$$(suc(t_1))_\alpha[2i] = (suc(cur))_\alpha[(t_1)_\alpha[2i]].$$

Otherwise, $(suc(t_1))_\alpha[2i] = \perp$.

- If $(t_1)_\alpha[2i] \neq \perp$, then

$$(pred(t_1))_\alpha[2i] = (pred(cur))_\alpha[(t_1)_\alpha[2i]].$$

Otherwise, $(pred(t_1))_\alpha[2i] = \perp$,

- If there exists $j : i < j \leq n$ such that $a_j \in A$ and j is the minimum number satisfying this condition, that is, for every $j' : i < j' < j$, we have $a_{j'} \notin A$, then $(suc_A(cur))_\alpha[2i] = 2j$. Otherwise, $(suc_A(cur))_\alpha[2i] = \perp$.
- If there exists $j : j \leq i$ such that $a_j \in A$ and j is the maximum number satisfying this condition, that is, for every $j' : j < j' \leq i$, we have $a_{j'} \notin A$, then $(pred_A(cur))_\alpha[2i] = 2(j - 1)$. Otherwise, $(pred_A(cur))_\alpha[2i] = \perp$.

- If $(t_1)_\alpha[2i] \neq \perp$, then

$$(suc_A(t_1))_\alpha[2i] = (suc_A(cur))_\alpha[(t_1)_\alpha[2i]].$$

Otherwise, $(suc_A(t_1))_\alpha[2i] = \perp$.

- If $(t_1)_\alpha[2i] \neq \perp$, then

$$(pred_A(t_1))_\alpha[2i] = (pred_A(cur))_\alpha[(t_1)_\alpha[2i]].$$

Otherwise, $(pred_A(t_1))_\alpha[2i] = \perp$.

EXAMPLE 1. Suppose

$$\alpha = \begin{array}{cccccccccccc} d_0 & a & d_1 & b & d_2 & a & d_3 & a & d_4 & b & d_5 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array},$$

where the second arrow is the sequence of positions. Let $t_1 = suc_a(suc(cur))$ and $t_2 = pred_b(pred(cur))$. Let us consider $(t_1)_\alpha[0]$ and $(t_2)_\alpha[10]$. At first, $(t_1)_\alpha[0] = (suc_a(cur))_\alpha[(suc(cur))_\alpha[0]]$. Since $(suc(cur))_\alpha[0] = 2$, and the first occurrence of a after the position 2 is in the position 5, we get $(t_1)_\alpha[0] = 6$. On the other hand, $(t_2)_\alpha[10] = (pred_b(cur))_\alpha[(pred(cur))_\alpha[10]]$. Because $(pred(cur))_\alpha[10] = 8$ and the last occurrence of b before the position 8 is in the position 3, we get $(t_2)_\alpha[10] = 2$.

A rigid data constraint c over Σ^\pm is defined by the following rules,

$$c := t_1 \sim t_2 \mid t_1 \not\sim t_2 \mid c \vee c \mid c \wedge c, \text{ where } t_1, t_2 \in \mathcal{T}_p[\Sigma^\pm].$$

We use $\mathcal{C}_{rgd}[\Sigma^\pm]$ to denote the set of rigid data constraints over Σ^\pm .

The semantics of rigid data constraints can be defined inductively. In the following, we will define the semantics for the case $c = t_1 \sim t_2$. The semantics of

$c = t_1 \not\sim t_2$ can be defined similarly. Moreover, the semantics of $c = c_1 \vee c_2$ and $c = c_1 \wedge c_2$ can be defined in a standard way. Let $c \in \mathcal{C}_{rgd}[\Sigma^\pm]$, $\alpha = d_0 a_1 d_1 \dots a_n d_n$, and $i : 0 \leq i \leq n$, then $(\alpha, 2i)$ is said to satisfy $c = t_1 \sim t_2$, denoted by $(\alpha, 2i) \models c$, if $(t_1)_\alpha[2i] \neq \perp$, $(t_2)_\alpha[2i] \neq \perp$, and $d_{(t_1)_\alpha[2i]} = d_{(t_2)_\alpha[2i]}$.

Given a rigid data constraint c , we use \bar{c} to denote the negation of c . More specifically, \bar{c} is obtained from c by swapping \sim for $\not\sim$, and \vee for \wedge . For instance, if $c = cur \sim suc_a(cur) \vee cur \not\sim pred(cur)$, then $\bar{c} = cur \not\sim suc_a(cur) \wedge cur \sim pred(cur)$.

PROPOSITION 1. The satisfiability problem of rigid data constraints is NP-complete.

3.2 Nondeterministic and deterministic rigid register automata

A nondeterministic rigid register automaton (NRRRA) \mathcal{A} over the alphabet Σ^\pm is a tuple (Q, δ, I, F) , where Q, I, F are as those in NRA, $\delta = \delta_w \cup \delta_d$ such that $\delta_w \subseteq Q_w \times \Sigma^\pm \times Q_d$ and $\delta_d \subseteq Q_d \times \mathcal{C}_{rgd}[\Sigma^\pm] \times Q_w$.

A run of \mathcal{A} over a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$ is a state sequence $q_0 c_0 q_1 a_1 q_2 \dots q_{2n-1} a_n q_{2n} c_n q_{2n+1}$ such that $q_0 \in I$, for every $i : 0 \leq i \leq n$, $(q_{2i}, c_i, q_{2i+1}) \in \delta_d$ and $(\alpha, 2i) \models c_i$, and for every $i : 1 \leq i \leq n$, $(q_{2i-1}, a_i, q_{2i}) \in \delta_w$. A run $\rho = q_0 c_0 q_1 a_1 q_2 \dots q_{2n-1} a_n q_{2n} c_n q_{2n+1}$ is accepting if $q_{2n+1} \in F$.

A deterministic rigid register automaton (DRRA) over Σ^\pm is a NRRRA $\mathcal{A} = (Q, \delta, I, F)$ such that I is a singleton, and δ satisfies that for every $q \in Q_w, a \in \Sigma^\pm$, there is at most one $q' \in Q_d$ such that $(q, a, q') \in \delta_w$, and for every $(q, c_1, q_1), (q, c_2, q_2) \in \delta_d$, if $q_1 \neq q_2$, then $c_1 \wedge c_2$ is unsatisfiable.

Let \mathcal{A} be a NRRRA. Then $\mathcal{T}_\mathcal{A}$ is used to denote the minimal set of position terms satisfying that for every $t_1 \sim t_2$ or $t_1 \not\sim t_2$ occurring in \mathcal{A} , we have $t_1, t_2 \in \mathcal{T}_\mathcal{A}$; moreover, if $t \in \mathcal{T}_\mathcal{A}$ and $t' \leq t$, then $t', t[t' \setminus cur] \in \mathcal{T}_\mathcal{A}$. In addition, $\mathcal{C}_\mathcal{A}$ is used to denote the set of rigid data constraints occurring in \mathcal{A} .

EXAMPLE 2. Let $\Sigma = \{a, b\}$. Let L denote the language of data paths satisfying that the sequence of word symbols on the data path belongs to ab^*a , the first data value occurs in some other position, and the last data value does not occur elsewhere. Then L is defined by the NRRRA \mathcal{A} illustrated in Figure 1, where suc_a is an abbreviation of $suc_a(cur)$, $Q_d = \{q_0, q_2, q_4, q_6\}$ and $Q_w = \{q_1, q_3, q_5, q_7, q_9, q_{11}\}$.

Since NRRAs are able to compare the current data value with the data values in the future, NRAs and NRRAs are expressively incomparable.

PROPOSITION 2. NRA and NRRRA are expressively incomparable.

Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA over the alphabet Σ^\pm and prj a letter projection from Σ^\pm to Γ . Then the letter projection of \mathcal{A} , denoted by $prj(\mathcal{A})$, is obtained

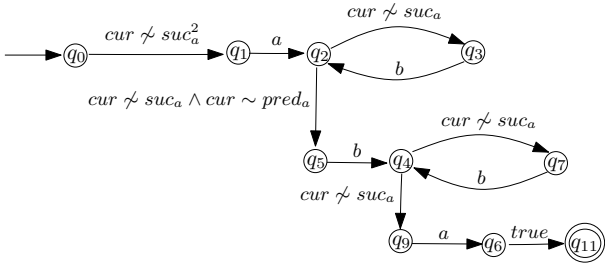


Figure 1: An example for NRRRA

from \mathcal{A} by replacing each transition $(q, a, q') \in \delta_w$ with $(q, prj(a), q')$, and each suc_A (resp. $pred_A$) occurring in δ_d , where $A \subseteq \Sigma^\pm$, with $suc_{prj(A)}$ (resp. $pred_{prj(A)}$). Note that $prj(\mathcal{A})$ may not define $prj(\mathcal{L}(\mathcal{A}))$, as witnessed by the following result.

PROPOSITION 3. *The class of languages definable by NRRAs are not closed under letter projections.*

In the following, we will introduce a constraint for a pair (\mathcal{A}, prj) , where \mathcal{A} is a NRRRA and prj is a letter projection, so that $prj(\mathcal{A})$ does define $prj(\mathcal{L}(\mathcal{A}))$.

Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA over Σ^\pm , prj be a letter projection from Σ^\pm to Γ . Then \mathcal{A} is said to be *position-invariant* under prj if for every suc_A (resp. $pred_A$) occurring in δ_d , where $A \subseteq \Sigma^\pm$, there is $B \subseteq \Gamma$ such that $A = prj^{-1}(B)$. It is easy to observe that the position-invariance guarantees that for every $t \in \mathcal{T}_\mathcal{A}$, every data path α and every position $2i$ of α , it holds $t_\alpha[2i] = (prj(t))_{prj(\alpha)}[2i]$, where $prj(t)$ is obtained from t by replacing each occurrence of suc_A (resp. $pred_A$) in t with $suc_{prj(A)}$ (resp. $pred_{prj(A)}$). From this, we deduce that if \mathcal{A} is position invariant under prj , then prj does not affect the interpretations of the rigid data constraints in \mathcal{A} . So we have the following result.

PROPOSITION 4. *Suppose \mathcal{A} is a NRRRA over Σ^\pm and prj is a letter projection from Σ^\pm to Γ . If \mathcal{A} is position-invariant under prj , then $\mathcal{L}(prj(\mathcal{A})) = prj(\mathcal{L}(\mathcal{A}))$.*

For a NRA, in every position, only a bounded number of data values occurring before this position are stored into the registers for the future references. On the other hand, in the first sight, in a NRRRA, it is only required that a bounded number of positions are referenced to by a data transition in a *single* position, but it is not required that only a bounded number of positions are referenced to by *all* the data transitions after a position. In the following, we show that this is indeed the case. By utilizing this property, we then show that NRRAs can be simulated by an extension of NRAs with nondeterministic guessing¹.

Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA over the alphabet Σ^\pm , $\alpha = d_0a_1d_1 \dots a_nd_n$ be a data path over Σ^\pm , $\rho =$

¹The idea of nondeterministic guessing, called nondeterministic reassignment, was introduced in [14].

$q_0c_0q_1a_1q_2 \dots q_{2n-1}a_nq_{2n}c_nq_{2n+1}$ be a run of \mathcal{A} over α , and $i : 0 \leq i \leq n$. Define the set of *future positions* of the position $2i$ of α with respect to ρ , denoted by $Pos_\rho^f[\alpha, 2i]$, as

$$\{t_\alpha[2j] \mid j \leq i, t \text{ occurs in } c_j, t_\alpha[2j] \neq \perp, t_\alpha[2j] > 2i\}.$$

Similarly, define the set of *past positions* of the position $2i$ of α with respect to ρ , denoted by $Pos_\rho^p[\alpha, 2i]$, as

$$\{t_\alpha[2j] \mid i \leq j, t \text{ occurs in } c_j, t_\alpha[2j] \neq \perp, t_\alpha[2j] < 2i\}.$$

LEMMA 1. *Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA over the alphabet Σ^\pm and $\alpha = d_0a_1d_1 \dots a_nd_n$ be a data path. Then for every run ρ of \mathcal{A} over α and every $i : 0 \leq i \leq n$, $Pos_\rho^f[\alpha, 2i] \cup Pos_\rho^p[\alpha, 2i] \subseteq \{t_\alpha[2i] \mid t \in \mathcal{T}_\mathcal{A}\}$.*

Intuitively, Lemma 1 says that for every run ρ over a data path α and every position $2i$ of α , only a bounded number of positions before (resp. after) the position $2i$ are referred to by ρ after (resp. before) reaching the position $2i$.

A *nondeterministic register data path automaton with guessing* (NRAG) \mathcal{A} over Σ^\pm is a tuple (Q, k, δ, I, F) , where Q, k, I, F are as those in the definition of NRA, $\delta \subseteq \delta_w \cup \delta_d$ such that $\delta_w \subseteq Q_w \times \Sigma^\pm \times Q_d$ and $\delta_d \subseteq Q_d \times \mathcal{C}_k \times Q_w \times 2^{[k]} \times 2^{[k]} \times \mathcal{C}_{2k}$ satisfies that for every $(q, c, q', X, Y, c') \in \delta_d$, it holds that $X \cap Y = \emptyset$, and c' does not contain r_{k+i} with $r_i \notin Y$.

The intuition of a transition $(q, c, q', X, Y, c') \in \delta$ is that if the current state is q , the data values stored in the registers together with the current data value d satisfies c , then the state is changed to q' , d is stored into every register in X . Meanwhile, for each register in Y , a data value is nondeterministically guessed. In addition, the guessed data values should satisfy the constraint c' .

The semantics of NRAGs are defined similarly as those of NRAs, that is, a successor relation of configurations $(q, j, \theta) \vdash_\alpha (q', j+1, \theta')$ can be defined, with the following adjustment for data transitions.

If the j -th symbol is a data value d , then there exist c, c' such that $(q, c, q', X, Y, c') \in \delta_d$, and θ' is obtained from θ as follows,

- for each $i \in X$, d is assigned to i (thus $\theta'(i) = d$),
- for each $i \in Y$, a data value d'_i is guessed (thus $\theta'(i) = d'_i$), so that the guessed data values satisfy the following condition: The function θ_g extending θ by assigning d'_i to $k+i$ for each $i \in Y$ satisfies that $(\theta_g, d) \models c'$,
- for each $i \notin X \cup Y$, $\theta'(i) = \theta(i)$.

Note that data values are not allowed to be copied explicitly among the registers in NRAG. But this can be achieved by guessing. For instance, if we want to *copy* a data value from r_i to r_j , then we can guess a data value for r_j and add the constraint $r_i \sim r_{k+j}$ for

the guessing. Later on, when we mention copying a data value from a register to the other, we always mean the implicit copying by guessing.

Since the nonemptiness of NRAGs can be solved similarly to that of NRAs, we have the following result.

PROPOSITION 5. *The nonemptiness problem of NRAG is PSPACE-complete.*

In the following, we will show that the expressive power of NRRAs can be captured by NRAGs.

THEOREM 3. *From a NRRRA $\mathcal{A} = (Q, \delta, I, F)$, an equivalent NRAG $\mathcal{B} = (Q', k, \delta', I', F')$ can be constructed such that $|Q'|$ is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$ and k is polynomial over $|\mathcal{T}_A|$.*

We will present a proof sketch for Theorem 3 and illustrate the main ideas. These ideas are also used for the proof of Theorem 6 in Section 5.

PROOF. Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA. In the following, we will construct a NRAG \mathcal{B} to simulate \mathcal{A} .

We first give an intuitive description of the construction. Let ρ be a run of \mathcal{A} over a data path α . Then in the position $2i$, \mathcal{B} simulates ρ as follows: \mathcal{B} records the data values in the positions belonging to $Pos_\rho^p[\alpha, 2i]$, guesses the data values in the positions belonging to $Pos_\rho^f[\alpha, 2i]$, and records the order for the positions in $Pos_\rho^p[\alpha, 2i]$ and $Pos_\rho^f[\alpha, 2i]$.

We introduce some additional notations.

Let $\alpha = d_0 a_1 d_1 \dots a_n d_n$ be a data path and $i : 0 \leq i \leq n$. The *profile* of the position $2i$ in α , denoted by $prof_\alpha(2i)$, is defined as a triple (S, χ, \sim) , where

- $S = \{t \in \mathcal{T}_A \mid t_\alpha[2i] \neq \perp\}$,
- χ is a sequence
 $(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})$
 $(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$

where

- for every $j : -m_1 \leq j \leq m_2$, $T_j \subseteq S$ and $T_j \neq \emptyset$,
- the collection $T_{-m_1}, \dots, T_0, \dots, T_{m_2}$ forms a partition of S , and $cur \in T_0$,
- for every $t, t' \in S$, if $t \in T_{j_1}$ and $t' \in T_{j_2}$, then $j_1 \leq j_2$ iff $t_\alpha[2i] \leq t'_\alpha[2i]$ (in particular, $j_1 = j_2$ iff $t_\alpha[2i] = t'_\alpha[2i]$),
- for every $j : -m_1 < j \leq m_2$, $b_j = a_{(t_\alpha[2i])/2}$ for some $t \in T_j$, and $b_{-m_1} = a_{(t_\alpha[2i])/2}$ if $t_\alpha[2i] > 0$ for some $t \in T_{-m_1}$, otherwise, $b_{-m_1} = \perp$,
- for every $j : -m_1 \leq j < m_2$, $b'_j = a_{(t_\alpha[2i])/2+1}$ for some $t \in T_j$, and $b'_{m_2} = a_{(t_\alpha[2i])/2+1}$ if $t_\alpha[2i] < 2n$ for some $t \in T_{m_2}$, otherwise, $b'_{m_2} = \perp$,
- $s_{m_2} = \perp$, and for every $j : -m_1 \leq j < m_2$, if $t'_\alpha(2i) = t_\alpha(2i) + 1$ for some $t \in T_j$ and $t' \in T_{j+1}$, then $s_j = 1$, otherwise, $s_j = 0$.

- \sim is an equivalence relation over S defined as follows: Let $t, t' \in S$, then $t \sim t'$ iff $d_{t_\alpha[2i]} = d_{t'_\alpha[2i]}$.

Let Σ_{prof} denote the set of all triples (S, χ, \sim) such that $S \subseteq \mathcal{T}_A$,

- χ is a sequence

$$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) \\ (b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

such that

- for every $j : -m_1 \leq j \leq m_2$, $T_j \neq \emptyset$,
- $cur \in T_0$, and T_{-m_1}, \dots, T_{m_2} is a partition of S ,
- $b_{-m_1} \in \Sigma^\pm \cup \{\perp\}$, and for every $j : -m_1 < j \leq m_2$, $b_j \in \Sigma^\pm$,
- $b'_{m_2} \in \Sigma^\pm \cup \{\perp\}$, and for every $j : -m_1 \leq j < m_2$, $b'_j \in \Sigma^\pm$,
- $s_{m_2} = \perp$, and for every $j : -m_1 \leq j < m_2$, $s_j \in \{0, 1\}$,

- \sim is an equivalence relation over S such that for every $t, t' \in \mathcal{T}_A$, if $t, t' \in T_j$ for some j , then $t \sim t'$.

Note that for $(S, \chi, \sim) \in \Sigma_{prof}$, there may be no data paths α and a position in α such that the profile of the position in α is (S, χ, \sim) . Nevertheless, we are able to define a consistency condition on the elements from Σ_{prof} so that a consistent element from Σ_{prof} indeed corresponds to the profile of a position in some data path. Moreover, for two consistent elements from Σ_{prof} , say $(S_1, \chi_1, \sim_1), (S_2, \chi_2, \sim_2)$, and $a \in \Sigma^\pm$, we are able to define a syntactic successor relation $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$, which mimics the changes from $prof_\alpha(2i)$ to $prof_\alpha(2(i+1))$ by reading a word symbol a in the position $2i+1$ of a data path. The details of the consistency condition and the successor relation are omitted due to the space limitation.

We are ready to construct the NRAG \mathcal{B} .

There are $2|\mathcal{T}_A| + 1$ registers in \mathcal{B} , that is,

$$r_1, \dots, r_{|\mathcal{T}_A|}, r_{|\mathcal{T}_A|+1}, \dots, r_{2|\mathcal{T}_A|}.$$

Over a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$, \mathcal{B} does the following.

- In each position $2i$ ($0 \leq i \leq n$), \mathcal{B} guesses $\pi_i = (S_i, \chi_i, \sim_i) \in Prof_A$ (where π_i is supposed to be $prof_\alpha[2i]$). In addition,
 - if $i = 0$, then $\pi_0 = (S_0, \chi_0, \sim_0)$ is an initial profile, that is, for every $t \in \mathcal{T}_A$ such that $pred(cur) \leq t$ or $pred_A(cur) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_0$,
 - if $i = n$, then $\pi_n = (S_n, \chi_n, \sim_n)$ is a final profile, that is, for every $t \in \mathcal{T}_A$ such that $suc(cur) \leq t$ or $suc_A(cur) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_n$.

- For every $i : 0 \leq i \leq n$, if

$$\chi_i = \begin{pmatrix} (b_{i,-m_{i,1}}, T_{i,-m_{i,1}}, b'_{i,-m_{i,1}}, s_{i,-m_{i,1}}) \dots \\ (b_{i,-1}, T_{i,-1}, b'_{i,-1}, s_{i,-1}) (b_{i,0}, T_{i,0}, b'_{i,0}, s_{i,0}) \dots \\ (b_{i,1}, T_{i,1}, b'_{i,1}, s_{i,1}) \dots \\ (b_{i,m_{i,2}}, T_{i,m_{i,2}}, b'_{i,m_{i,2}}, s_{i,m_{i,2}}) \end{pmatrix},$$

then after the position $2i$ is visited (that is, the reading head is in $2i+1$), for each $j : -m_{i,1} \leq j \leq m_{i,2}$, \mathcal{B} stores in the register $r_{j+|\mathcal{T}_A|}$ the data value corresponding to $T_{i,j}$. In particular, \mathcal{B} stores the data value d_i in $r_{|\mathcal{T}_A|}$.

- Over each pair of positions $2i$ and $2(i+1)$ (where $0 \leq i < n$), \mathcal{B} checks that $\pi_i \xrightarrow{a_{i+1}} \pi_{i+1}$. To do this, \mathcal{B} copies (by guessing) data values between registers and guesses some data values for a few registers.
- At the same time, \mathcal{B} simulates the run of \mathcal{A} as follows.
 - If \mathcal{A} makes a transition (q, a_i, q') over a_i , then \mathcal{B} checks that $b'_{i-1,0} = a_i$ and changes the state from q to q' .
 - If \mathcal{A} makes a transition (q, c, q') over d_i , then \mathcal{B} checks that π_i satisfies c , verifies that d_i is equal to the data value stored in $r_{j+|\mathcal{T}_A|}$ for each $j : -m_1 \leq j \leq m_2$ such that there is $t \in T_j$ satisfying $cur \sim_i t$ (in particular, d_i should be equal to the data value in $r_{|\mathcal{T}_A|}$), and changes the state from q to q' .
 - \mathcal{B} accepts if \mathcal{A} accepts and a final profile is reached.

From the above construction, we know that in its states, \mathcal{B} should record the states of \mathcal{A} and the guessed profiles. Therefore, the number of states of \mathcal{B} is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. \square

PROPOSITION 6. *The nonemptiness of NRRAs and DRRAs is PSPACE-complete.*

By using a slight extension of the subset construction, we are able to show that NRRAs can be determinized.

PROPOSITION 7. *For every NRRAs \mathcal{A} , there is an equivalent DRRAs of exponential size.*

COROLLARY 1. *NRRAs are closed under all Boolean operations.*

COROLLARY 2. *The language inclusion problem of NRRAs is PSPACE-complete.*

3.3 Two-way nondeterministic rigid register automata

In this subsection, we will show that two-way nondeterministic rigid register automata are of the same expressibility as NRRAs.

A two-way nondeterministic rigid register automaton (2NRRAs) \mathcal{A} over Σ^\pm is a tuple $(Q, \vdash, \neg, \delta, I, F)$, where Q, I, F are as those in the definition of NRRAs, $\vdash, \neg \in \Sigma^\pm$ are respectively the left and right endmarkers, $\delta = \delta_w \cup \delta_d$ such that

- $\delta_w \subseteq Q \times (\Sigma^\pm \cup \{\vdash, \neg\}) \times Q \times \{+1, -1\}$ (where $+1, -1$ denote the direction of the head: “right” and “left”) satisfies that for every transition $(q, \vdash, q', dir) \in \delta_w$ (resp. $(q, \neg, q', dir) \in \delta_w$), it holds that $dir = +1$ (resp. $dir = -1$),

- $\delta_d \subseteq Q \times \mathcal{C}_{rgd} \times Q \times \{+1, -1\}$.

Let $\alpha = d_0 a_1 d_1 \dots a_n d_n$ be a data path and \mathcal{A} be a 2NRRAs. A run of \mathcal{A} over α is a sequence

$$(q_0, i_0) \theta_0 (q_1, i_1) \theta_1 \dots \theta_{m-1} (q_m, i_m)$$

such that $q_0 \in I$, $i_0 = 0$, $i_m = 2n + 2$,

- for every $j : 0 \leq j < m$, if the symbol of $\vdash \alpha \neg$ in the position i_j is a word symbol $a \in \Sigma^\pm \cup \{\vdash, \neg\}$, then there is $dir \in \{+1, -1\}$ such that $(q_j, a, q_{j+1}, dir) \in \delta_w$, $\theta_j = a$, and $i_{j+1} = i_j + dir$,
- for every $j : 0 \leq j \leq m$, if the symbol of $\vdash \alpha \neg$ in the position i_j is a data value d , then there are $c \in \mathcal{C}_{rgd}$ and $dir \in \{+1, -1\}$ such that $(q_j, c, q_{j+1}, dir) \in \delta_d$, $(\alpha, i_j - 1) \models c$, $\theta_j = c$, and $i_{j+1} = i_j + dir$.

A run is accepting if $q_m \in F$. Note that a run of a 2NRRAs over α starts at the left endmarker (position 0) and stops at the right endmarker (position $2n + 2$).

PROPOSITION 8. *For every 2NRRAs, there is an equivalent NRRAs of exponential size.*

3.4 Rigid regular expressions with memory

Rigid regular expressions with memory (RREM) is defined by the following rules,

$$e := \varepsilon \mid a \mid [c] \mid e \cup e \mid e \cdot e \mid e^+, \text{ where } c \in \mathcal{C}_{rgd}[\Sigma^\pm].$$

Let e be a RREM, $\alpha = d_0 a_1 d_1 \dots a_n d_n$ be a data path, and $i, j : 0 \leq i \leq j \leq 2n$. The semantics of e is defined by a relation $(\alpha, i) \vdash_e (\alpha, j)$ as follows.

- If $e = \varepsilon$, then $(\alpha, i) \vdash_e (\alpha, j)$ if $i = j$ and the symbol of α at position i is a data value (thus i is even).
- If $e = a$, then $(\alpha, i) \vdash_e (\alpha, j)$ if $j = i + 2$, the symbol of α at position $i + 1$ is a .
- If $e = [c]$, then $(\alpha, i) \vdash_e (\alpha, j)$ if $i = j$, the symbol of α at position i is a data value (thus i is even), and $(\alpha, i) \models c$.
- The semantics for the rules $e_1 \cup e_2$, $e_1 \cdot e_2$ and e_1^+ are defined in a natural way and are omitted.

A data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$ is accepted by a RREM e if $(\alpha, 0) \vdash_e (\alpha, 2n)$. Let $\mathcal{L}(e)$ denote the set of data paths accepted by a RREM e .

PROPOSITION 9. *NRRAs and RREMs have the same expressive power.*

- From a RREM e , a NRRAs \mathcal{A}_e can be constructed in LOGSPACE such that $\mathcal{L}(e) = \mathcal{L}(\mathcal{A}_e)$.

- From a NRRA \mathcal{A} , a RREM $e_{\mathcal{A}}$ can be constructed in EXPTIME such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(e_{\mathcal{A}})$.

COROLLARY 3. *The nonemptiness problem of RREMs is PSPACE-complete.*

4. RIGID REGULAR PATH QUERIES WITH DATA

A rigid regular path query with inverse and data (2RRDPQ) ξ over the alphabet Σ is a tuple (x, L, y) where L is a language of data paths defined by a NRRA or a RREM over Σ^\pm .

Given a data graph $\mathcal{G} = (V, E, \eta)$ and a RRDPQ $\xi = (x, L, y)$, the evaluation result of ξ over \mathcal{G} , denoted by $\xi(\mathcal{G})$, is the set of all pairs (v, v') such that there is a semipath π from v to v' in \mathcal{G} such that $\eta(\pi) \in L$.

PROPOSITION 10. *The evaluation problem of 2RRDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.*

In the following we will show that every 2RDPQ can be turned into a 2RRDPQ, if data graphs are transformed in a natural way. Note that the transformation of data graphs presented in the following is *localized* in the sense that for each node, a new node is added and connected to the node by edges with special labels, and the relationships between the nodes in the original data graph are not changed.

Let $\mathcal{G} = (V, E, \eta)$ be a data graph over the alphabet Σ , $k \geq 1$, and $\{A_i \mid 1 \leq i \leq k\} \cap \Sigma = \emptyset$. Then the *data-to-node k -transformation* of \mathcal{G} , denoted by $\mathcal{G}_{dn,k} = (V_{dn,k}, E_{dn,k}, \eta_{dn,k})$, is defined as follows.

- $V_{dn,k}$ is obtained from V by adding a new node n_v for each node $v \in V$,
- $E_{dn,k}$ is defined as the union of E and the set of edges (v, A_i, n_v) for every $v \in V$ and $i : 1 \leq i \leq k$,
- for each $v \in V$, $\eta_{dn,k}(v) = \eta(v)$ and $\eta_{dn,k}(n_v) = \eta(v)$.

The intuition of the above transformation is to copy the data value of each node v to a new node connected to v with k edges. Note that the transformation does not change the edges between nodes in the original graph.

THEOREM 4. *Let $k \geq 1$, $\xi = (x, L, y)$ be a 2RDPQ over the alphabet Σ such that L is given by a NRA or REM containing at most k -registers. Then a 2RRDPQ $\xi' = (x, L', y)$ over the alphabet $\Sigma^\pm \cup \{A_i, A_i^- \mid 1 \leq i \leq k\}$ can be constructed in polynomial time such that for every data graph $\mathcal{G} = (V, E, \eta)$, $\xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn,k})$.*

Note that in practice, the number k in 2RDPQs are usually small, e.g. $k = 1, 2$, and can be assumed to be a constant. Then the above data-to-node transformation becomes query-independent.

5. CONJUNCTIVE RIGID REGULAR PATH QUERIES WITH DATA

Conjunctive 2RRDPQs (C2RRDPQ) can be defined similarly to C2RDPQs, with 2RDPQs replaced by 2RRDPQs.

PROPOSITION 11. *The evaluation of C2RRDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.*

THEOREM 5. *The containment of C2RRDPQs is in 2EXPSpace and EXPSpace hard.*

The rest of this section is devoted to the proof of Theorem 5. The proof is a nontrivial extension of that of the EXPSpace-completeness result for C2RDPQs in [8] and is the most technical part of this paper.

5.1 Canonical data graph

Let $\xi := \text{Ans}(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$ be a C2RRDPQ, $\mathcal{G} = (V, E, \eta)$ be a data graph, and $\nu : \{y_1, \dots, y_{2l}\} \rightarrow V$. Then \mathcal{G} is said to be ν -canonical for ξ if

- \mathcal{G} consists of l simple semipaths π_1, \dots, π_l , one for each conjunct of ξ , such that only start and end nodes can be shared among different semipaths.
- for every $i : 1 \leq i \leq l$, π_i is a semipath from $\nu(y_{2i-1})$ to $\nu(y_{2i})$ such that $\eta(\pi_i)$ belongs to L_i .

It is easy to see that if \mathcal{G} is ν -canonical for ξ , then $\nu(\bar{z})$ belongs to $\xi(\mathcal{G})$.

In the rest of this section, we assume that ξ_1, ξ_2 are two C2RRDPQs such that

- ξ_1 and ξ_2 have the same set of distinguished variables,
- the set of non-distinguished variables of ξ_1 and ξ_2 are disjoint.

More specifically, for $i = 1, 2$, let

$$\xi_i := \text{Ans}(z_1, \dots, z_n) \leftarrow \bigwedge_{1 \leq j \leq l_i} (y_{i,2j-1}, L_{i,j}, y_{i,2j})$$

such that $\{y_{1,1}, \dots, y_{1,2l_1}\} \cap \{y_{2,1}, \dots, y_{2,2l_2}\}$ is equal to $\{z_1, \dots, z_n\}$.

Let $\mathcal{G} = (V, E, \eta)$ be a ν -canonical data graph for ξ_1 . Then a mapping $\mu : \{y_{2,1}, \dots, y_{2,2l_2}\} \rightarrow V$ is said to be a $(\xi_1, \mathcal{G}, \nu)$ -mapping for ξ_2 if

- for every $j : 1 \leq j \leq n$, $\nu(z_j) = \mu(z_j)$,
- for every $j : 1 \leq j \leq l_2$, $(\mu(y_{2,2j-1}), \mu(y_{2,2j}))$ belongs to the evaluation result of $(y_{2,2j-1}, L_{2,j}, y_{2,2j})$ over \mathcal{G} .

Note that the existence of a $(\xi_1, \mathcal{G}, \nu)$ -mapping for ξ_2 implies that $\nu(\bar{z}) \in \xi_2(\mathcal{G})$.

The following result can be shown in the same way as a corresponding result for C2RDPQs (Theorem 2 in [8]).

PROPOSITION 12. Let ξ_1, ξ_2 be two C2RRDPQs. Then $\xi_1 \not\subseteq \xi_2$ iff there are a data graph \mathcal{G} and a mapping ν from the variables in ξ_1 to the nodes in \mathcal{G} such that

- \mathcal{G} is ν -canonical for ξ_1 ,
- and there are no $(\xi_1, \mathcal{G}, \nu)$ -mappings for ξ_2 .

5.2 Evaluating 2RRDPQs over canonical data graphs

Let $\mathcal{G} = (V, E, \eta)$ be a ν -canonical data graph for ξ_1 and $\xi = (x, L, y)$ be a 2RRDPQ such that L is defined by a NRRA $\mathcal{A} = (Q, \delta, I, F)$ over Σ^\pm . Then \mathcal{G} consists of l_1 -simple semipaths π_1, \dots, π_{l_1} such that for every $j : 1 \leq j \leq l_1$, π_j is a semipath from $\nu(y_{1,2j-1})$ to $\nu(y_{1,2j})$, and $\eta(\pi_j) \in L_{1,j}$. Our goal is to evaluate ξ over \mathcal{G} .

We use a similar idea to the evaluation of 2RPQs over canonical graphs in [8]: the data graph \mathcal{G} is first encoded into a data path $\alpha_{\mathcal{G}}$, then a 2NRRA \mathcal{A}_ξ is constructed from ξ and ξ_1 so that $\xi(\mathcal{G})$ is nonempty iff $\vdash \alpha_{\mathcal{G}} \dashv$ is accepted by \mathcal{A}_ξ .

For every $i : 1 \leq i \leq l_1$, let ren_i denote the renaming function that maps each $a \in \Sigma^\pm$ to (a, i) . For a data path α , let $\text{ren}_i(\alpha)$ denote the data path obtained from α by replacing each $a \in \Sigma^\pm$ with $\text{ren}_i(a)$.

Let $\Sigma_{\xi_1} = \{\#\} \cup \bigcup_{1 \leq j \leq l_1} (\Sigma^\pm \times \{j\} \cup \{\$2_{j-1}, \$2_j\})$. We represent \mathcal{G} as a data path $\alpha_{\mathcal{G}}$ over the alphabet Σ_{ξ_1} as follows.

$$\alpha_{\mathcal{G}} := d_1 \$1 \text{ren}_1(\eta(\pi_1)) \$2 d_2 \# d_3 \$3 \text{ren}_2(\eta(\pi_2)) \$4 d_4 \# \dots d_{2l_1-1} \$2_{l_1-1} \text{ren}_{l_1}(\eta(\pi_{l_1})) \$2_{l_1} d_{2l_1},$$

where $d_1, d_2, \dots, d_{2l_1}$ are data values from \mathbb{D} not occurring in \mathcal{G} such that $d_i = d_j$ iff $\nu(y_{1,i}) = \nu(y_{1,j})$. Intuitively, for each $j : 1 \leq j \leq l_1$, the j -th semipath π_j is represented by a data subpath $\alpha_{\mathcal{G}, \pi_j}$ in $\alpha_{\mathcal{G}}$, where $\alpha_{\mathcal{G}, \pi_j} = d_{2j-1} \$2_{j-1} \text{ren}_j(\eta(\pi_j)) \$2_j d_{2j}$, and the symbol $\#$ is used to separate those data subpaths. It is easy to observe that for every pair (π_j, v) such that v is a node in π_j , there is a unique position in $\alpha_{\mathcal{G}}$ corresponding to (π_j, v) , denoted by $p_{\alpha_{\mathcal{G}}}(\pi_j, v)$. For instance, if $v = \nu(y_{1,2j}) = \nu(y_{1,2j'-1})$, then $p_{\alpha_{\mathcal{G}}}(\pi_j, v)$ is the position immediately before the symbol $\$2_j$ and $p_{\alpha_{\mathcal{G}}}(\pi_{j'}, v)$ is the position immediately after $\$2_{j'-1}$ in $\alpha_{\mathcal{G}}$.

For the simplicity of presentations, we assume that for every $j : 1 \leq j \leq l_1$, π_j contains at least two edges. All the proofs in the rest of this section can be easily adapted to deal with the situation that there is $j : 1 \leq j \leq l_1$ such that π_j contains at most one edge.

Let $\pi' = v_0 a_1 v_1 \dots v_{\ell-1} a_\ell v_\ell$ be a semipath in \mathcal{G} (since π' is an arbitrary semipath in \mathcal{G} , it may start or end in the middle of π_1, \dots, π_{l_1}). Because $\alpha_{\mathcal{G}}$ is an encoding of the data graph \mathcal{G} and π' is a semipath in \mathcal{G} , there is also an encoding of π' in $\alpha_{\mathcal{G}}$. We call this encoding as the *trace* of π' in $\alpha_{\mathcal{G}}$, denoted by $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$. A formal definition of $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$ will be given later.

The intuition of the 2NRRA \mathcal{A}_ξ is that for every semipath π' of \mathcal{G} and every run of \mathcal{A} over $\eta(\pi')$, \mathcal{A}_ξ goes through the trace of π' in $\alpha_{\mathcal{G}}$ to simulate the run of \mathcal{A} over $\eta(\pi')$.

THEOREM 6. Let \mathcal{G} be a ν -canonical data graph for ξ_1 , ξ be a 2RRDPQ. Then a 2NRRA \mathcal{A}_ξ can be constructed from ξ and ξ_1 such that $\xi(\mathcal{G})$ is nonempty iff \mathcal{A}_ξ accepts $\vdash \alpha_{\mathcal{G}} \dashv$.

In the following, before giving a proof for Theorem 6, we first give the definition of traces of semipaths of \mathcal{G} in $\alpha_{\mathcal{G}}$, then state and prove an important lemma.

Let $\pi' = v_0 a_1 v_1 \dots v_{\ell-1} a_\ell v_\ell$ be a semipath in \mathcal{G} . The $\bar{\pi}$ -unraveling (where $\bar{\pi} = (\pi_1, \dots, \pi_{l_1})$) of π' , denoted by $\text{urv}_{\bar{\pi}}(\pi')$, is defined as the sequence $\pi'_0 \# \pi'_1 \# \dots \# \pi'_r$ satisfying the following conditions: There are i_0, \dots, i_{r+1} such that

- $0 = i_0 < i_1 < \dots < i_r < i_{r+1} = \ell$,
- for every $0 \leq s \leq r$, $\pi'_s = v_{i_s} a_{i_s+1} v_{i_s+1} \dots a_{i_{s+1}} v_{i_{s+1}}$,
- for every $s : 0 \leq s \leq r$, there is $j_s : 1 \leq j_s \leq l_1$ such that all the edges on π'_s belong to π_{j_s} ,
- and for every $s : 1 \leq s \leq r$, either $j_s \neq j_{s-1}$, or $j_s = j_{s-1}$ and one of the following conditions holds,
 - the last edge of π'_{s-1} is the first edge of π_{j_s} and the first edge of π'_s is the last edge of π_{j_s} ,
 - the last edge of π'_{s-1} is the last edge of π_{j_s} and the first edge of π'_s is the first edge of π_{j_s} .

The last two conditions above correspond to the situation that the two endpoints of π_{j_s} are in fact the same node and a semipath can jump from the first (resp. last) edge to the last (resp. first) edge of π_{j_s} .

For a semipath π' in \mathcal{G} , define $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$, the trace of π' in $\alpha_{\mathcal{G}}$, as $\text{trc}_{\alpha_{\mathcal{G}}}(\pi'_0) \# \dots \# \text{trc}_{\alpha_{\mathcal{G}}}(\pi'_r)$, where $\pi'_0 \# \dots \# \pi'_r$ is the $\bar{\pi}$ -unraveling of π' , and for every $s : 0 \leq s \leq r$,

$$\text{trc}_{\alpha_{\mathcal{G}}}(\pi'_s) := \begin{matrix} p_{\alpha_{\mathcal{G}}}(\pi_{j_s}, v_{i_s}) & a_{i_s+1} & p_{\alpha_{\mathcal{G}}}(\pi_{j_s}, v_{i_{s+1}}) \\ \dots & a_{i_{s+1}} & p_{\alpha_{\mathcal{G}}}(\pi_{j_s}, v_{i_{s+1}}) \end{matrix}.$$

Note that although $\text{urv}_{\bar{\pi}}(\pi')$ and $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$ are not data paths, they are of a similar structure, that is, nodes and position indices respectively separated by word symbols.

For brevity, later on, when $\alpha_{\mathcal{G}}$ is obvious from the context, we abbreviate $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$ as $\text{trc}(\pi')$.

It is easy to see that a run of \mathcal{A} over $\eta(\pi')$ for a semipath π' in \mathcal{G} can be transformed into a run of a NRRA \mathcal{A}' over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$, if the interpretation of position terms over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$ is adjusted to jump over the additional $\#$ symbols as follows.

Since $\eta(\text{urv}_{\bar{\pi}}(\pi')) = \eta(\pi'_0) \# \dots \# \eta(\pi'_r)$, it follows that $\eta(\text{urv}_{\bar{\pi}}(\pi')) = d_0 b_1 d_1 \dots b_{\ell+2r} d_{\ell+2r}$ for $d_0, \dots, d_{\ell+2r} \in \mathbb{D}$ and $b_1, \dots, b_{\ell+2r} \in \Sigma^\pm \cup \{\#\}$. For a position term $t \in$

$\mathcal{T}_p[\Sigma^\pm]$ and a position $2i : 0 \leq i \leq \ell + 2r$ on $\eta(\text{urv}_{\bar{\pi}}(\pi'))$, define the *adjusted position* represented by t over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$ and $2i$, denoted by $t_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i]$, similarly to the semantics of position terms, with the following adjustments for the rules $\text{suc}(t_1)$ and $\text{pred}(t_1)$. In the following, we only present the adjustments for $\text{suc}(t_1)$, and the adjustments for $\text{pred}(t_1)$ are symmetric. If $(t_1)_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = \perp$ or $(t_1)_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = 2(\ell + 2r)$, then $(\text{suc}(t_1))_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = \perp$; otherwise,

- if $(t_1)_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i]$ is not a position immediately before $\#$, then

$$(\text{suc}(t_1))_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = (t_1)_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] + 2,$$

- otherwise,

$$(\text{suc}(t_1))_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = (t_1)_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] + 4.$$

LEMMA 2. *Suppose $\pi' = v_0 a_1 v_1 \dots a_\ell v_\ell$ is a semipath in \mathcal{G} such that $\text{urv}_{\bar{\pi}}(\pi') = \pi'_0 \# \dots \# \pi'_r$ and $\text{trc}(\pi') = p_0 b_1 p_1 \dots b_{\ell+2r} p_{\ell+2r}$ (where $b_1, \dots, b_{\ell+2r} \in \Sigma^\pm \cup \{\#\}$). Then for every $i : 0 \leq i \leq \ell + 2r$, there exists a function $\text{pos}_i \in (\mathcal{T}_p[\Sigma_{\xi_1}] \cup \{\perp\})^{\mathcal{T}_A}$ such that for every $t \in \mathcal{T}_A$, $\text{pos}_i(t) = \perp$ iff $t_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = \perp$; moreover, if $\text{pos}_i(t) \neq \perp$ and $t_{\text{urv}_{\bar{\pi}}(\pi')}^{\text{adj}}[2i] = 2i'$, then $(\text{pos}_i(t))_{\alpha_{\mathcal{G}}}[p_i] = p_{i'}$.*

Lemma 2 establishes a connection between the position terms in $\mathcal{T}_p[\Sigma]$ interpreted over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$ and the position terms in $\mathcal{T}_p[\Sigma_{\xi_1}]$ interpreted over $\alpha_{\mathcal{G}}$. With this connection, a 2NRRRA \mathcal{B} can be constructed such that each run of \mathcal{A}' over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$ can be simulated by a run of \mathcal{B} over $\text{trc}(\pi')$ in $\alpha_{\mathcal{G}}$.

PROOF. (Theorem 6)

Let π' be a path in \mathcal{G} , the $\bar{\pi}$ -unraveling of π be $\pi'_0 \dots \pi'_r$. In addition, for every $s : 0 \leq s \leq r$, all the edges on $\pi'_s = v_{i_s} a_{i_s+1} v_{i_s+1} \dots v_{i_{s+1}}$ belong to π_{j_s} .

Our goal is to construct a 2NRRRA \mathcal{B} over $\alpha_{\mathcal{G}}$ to simulate the runs of \mathcal{A}' over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$.

Similarly to the construction of NRAGs from NR-RAs in the proof of Theorem 3, the 2NRRRA \mathcal{B} goes through $\text{trc}(\pi')$ in $\alpha_{\mathcal{G}}$ and guesses the profile of the current position of $\eta(\text{urv}_{\bar{\pi}}(\pi'))$, in order to simulate \mathcal{A}' over $\eta(\text{urv}_{\bar{\pi}}(\pi'))$. The difference is that instead of storing and guessing the data values, \mathcal{B} records and guesses position terms from $\mathcal{T}_p[\Sigma_{\xi_1}]$ (interpreted over $\alpha_{\mathcal{G}}$) for position terms occurring in the profile of the current position in $\eta(\text{urv}_{\bar{\pi}}(\pi'))$. The most technical part of the construction is how to guarantee the consistency of the guessed position terms from $\mathcal{T}_p[\Sigma_{\xi_1}]$ and how to update them during the simulation. Since the details of the consistency conditions and the updating of the guessed position terms are rather tedious, they are omitted due to the space limitation.

From the above description, we know that in its states, \mathcal{B} should record the states of \mathcal{A}' , the guessed profiles,

and the guessed position terms from $\mathcal{T}_p[\Sigma_{\xi_1}]$. Because both the number of profiles and the number of possible guesses for the position terms from $\mathcal{T}_p[\Sigma_{\xi_1}]$ are exponential over $|\mathcal{T}_A|$, it follows that the number of states of \mathcal{B} is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. \square

5.3 Checking the non-containment

We will construct a NRRRA $\mathcal{A}' = (Q', \delta', I', F')$ to check the non-containment of ξ_1 over ξ_2 as follows.

1. Construct a NRRRA \mathcal{A}'_1 which reads a data path α over the alphabet Σ_{ξ_1} and verifies that α encodes a ν -canonical data graph \mathcal{G} for ξ_1 . In particular, for every 2RRDPQ $(y_{1,2j-1}, L_{1,j}, y_{1,2j})$, \mathcal{A}'_1 checks that the j -th block of α encodes a data path over the alphabet Σ^\pm belonging to $L_{1,j}$.
2. Construct a NRRRA \mathcal{A}'_2 verifying that there are no $(\xi_1, \mathcal{G}, \nu)$ -mappings for ξ_2 as follows.
 - (a) Construct a 2NRRRA \mathcal{B}_1 to verify a $(\xi_1, \mathcal{G}, \nu)$ -mapping for ξ_2 over $\alpha_{\mathcal{G}}$ annotated with subsets of $\{y_{2,1}, \dots, y_{2,l_2}\}$. The intention is that the annotations encode an assignment of nodes in \mathcal{G} to the variables from $\{y_{2,1}, \dots, y_{2,l_2}\}$. The alphabet of \mathcal{B}_1 is $\Sigma_{\xi_1}^e = \Sigma_{\xi_1} \times 2^{\{y_{2,1}, \dots, y_{2,l_2}\}}$. If the word symbol immediately before a position $2i$ of the annotated $\alpha_{\mathcal{G}}$ is (a', Z) , then this means that each variable in Z is assigned to the node of \mathcal{G} represented by the position $2i$. Some consistency constraints for these annotations, e.g. the annotations in two distinct positions are disjoint, should be checked. To check the 2RRDPQs $(y_{2,2j-1}, L_{2,j}, y_{2,2j})$ of ξ_2 over the annotated $\alpha_{\mathcal{G}}$, the construction in the proof of Theorem 6 is used. Note that since all the rigid data constraints in the RRDPQs of ξ_2 are independent from the annotations, we are able to assume that for every suc_B or pred_B occurring in \mathcal{B}_1 , there is $A \subseteq \Sigma_{\xi_1}$ such that $B = A \times 2^{\{y_{2,1}, \dots, y_{2,l_2}\}}$.
 - (b) Transform \mathcal{B}_1 into an equivalent NRRRA \mathcal{B}_2 (cf. Proposition 8).
 - (c) Let $\text{prj} : \Sigma_{\xi_1}^e \rightarrow \Sigma_{\xi_1}$ such that $\text{prj}((a', Z)) = a'$. Construct $\mathcal{B}_3 = \text{prj}(\mathcal{B}_2)$. From the assumption above, we know that \mathcal{B}_1 and \mathcal{B}_2 are position-invariant under prj . Then from Proposition 4, we deduce that $\mathcal{L}(\mathcal{B}_3) = \mathcal{L}(\text{prj}(\mathcal{B}_2)) = \text{prj}(\mathcal{L}(\mathcal{B}_2))$. So the NRRRA \mathcal{B}_3 guesses and verifies a $(\xi_1, \mathcal{G}, \nu)$ -mapping for ξ_2 .
 - (d) Determinize and complement \mathcal{B}_3 to get \mathcal{A}'_2 (cf. Proposition 7).
3. \mathcal{A}' is the intersection of \mathcal{A}'_1 and \mathcal{A}'_2 .

There is a final remark for the above construction: As pointed out in [8], letter projections are only meaningful

for one-way automata. This explains why we need go from the 2NRRAs \mathcal{B}_1 to the NRRAs \mathcal{B}_2 before applying the letter projection pr_j .

The complexity analysis.

The size of \mathcal{A}'_1 is polynomial over the size of ξ_1 . From Theorem 6, the size of \mathcal{B}_1 is exponential over the size of ξ_2 . From Proposition 8, the size of \mathcal{B}_2 is exponential over the size of \mathcal{B}_1 . The size of \mathcal{B}_3 is the same as the size of \mathcal{B}_2 . From Proposition 7, the size of \mathcal{A}'_2 is exponential over the size of \mathcal{B}_3 . Therefore, the size of \mathcal{A}'_2 is triple-exponential over the size of ξ_2 .

To check the nonemptiness of $\mathcal{A}'_1 \cap \mathcal{A}'_2$, we can guess “on the fly” an accepting run of $\mathcal{A}'_1 \cap \mathcal{A}'_2$ in double exponential space. From Savitch’s theorem, we deduce that the containment of C2RRDPQs is in 2EXPSpace.

On the other hand, the containment of C2RRDPQs is EXPSpace-hard since this is already the case for C2RPQs ([8]).

6. CONCLUSION

In this paper, a novel approach to extend 2RPQs with data value comparisons, called rigid regular path queries with inverse and data (2RRDPQs), was proposed. 2RRDPQs rely on nondeterministic rigid register automata (NRRAs), also introduced in this paper. We demonstrated the robustness of NRRAs by showing that NRRAs can be determinized and the two-way NRRAs are expressively equivalent to NRRAs. We then argued that 2RRDPQs achieve a good balance between the expressibility and computational properties. On the one hand, we showed that every 2RDPQ can be turned into a 2RRDPQ if a localized transformation is applied to graph databases. On the other hand, we proved that 2RRDPQs enjoy nice computational properties, as witnessed by the decidability (as a matter of fact, 2EXPSpace) of the containment problem of 2RRDPQs and conjunctive 2RRDPQs (C2RRDPQ). The proof for the 2EXPSpace result of the containment problem of C2RRDPQs is the most technical part of this paper and can be seen as the main result of this paper.

There are several natural directions for future work. One direction is to investigate whether the evaluation and containment problem of acyclic C2RRDPQs have a lower complexity. Another direction is to investigate nested rigid regular expressions with memory.

7. REFERENCES

- [1] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211(1-2):253–273, 1999.
- [2] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008.
- [3] P. Barceló. Querying graph databases. In *PODS*, pages 175–188, 2013.
- [4] P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations and the generalized intersection problem. In *LICS*, pages 115–124, 2012.
- [5] P. Barceló, C. Hurtado, L. Libkin, and P. Wood. Expressive languages for path queries over graph-structured data. In *PODS*, pages 3–14, 2010.
- [6] P. Barceló, J. Pérez, and J. L. Reutter. Relative expressiveness of nested regular expressions. In *AMW*, pages 180–195, 2012.
- [7] M. Benedikt, C. Ley, and G. Puppis. Automata vs. logics on data words. In *CSL*, pages 110–124, 2010.
- [8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR*, pages 176–185, 2000.
- [9] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A graphical query language supporting recursion. *SIGMOD Rec.*, 16(3):323–330, 1987.
- [10] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Logic*, 10(3):16:1–16:30, 2009.
- [11] A. Deutsch and V. Tannen. Optimization properties for classes of conjunctive regular path queries. In *DBPL*, pages 21–39, 2002.
- [12] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *PODS*, pages 139–148, 1998.
- [13] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.
- [14] M. Kaminski and D. Zeitlin. Extending finite-memory automata with non-deterministic reassignment (extended abstract). In *AFL*, pages 195–207, 2008.
- [15] E. V. Kostylev, J. L. Reutter, and D. Vrgoč. Regular path queries on graphs with data. In *ICDT*, 2014. To appear.
- [16] L. Libkin, J. Reutter, and D. Vrgoč. Trial for rdf: Adapting graph query languages for rdf data. In *PODS*, pages 201–212, 2013.
- [17] L. Libkin and D. Vrgoč. Regular expressions for data words. In *LPAR*, pages 274–288, 2012.
- [18] L. Libkin and D. Vrgoč. Regular path queries on graphs with data. In *ICDT*, pages 74–85, 2012.
- [19] J. Pérez, M. Arenas, and C. Gutierrez. nSPARQL: A navigational language for rdf. *Web Semant.*, 8(4):255–270, 2010.
- [20] J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3(2):198–200, 1959.
- [21] P. T. Wood. Query languages for graph databases. *SIGMOD Rec.*, 41(1):50–60, 2012.

APPENDIX

A. PROOFS IN SECTION 3.1

Proposition 1 *The satisfiability problem of rigid data constraints is NP-complete.*

PROOF. Lower bound: By an easy reduction from the satisfiability of Boolean formulas.

Upper bound: Let c be a rigid data constraint over Σ^\pm .

Let \mathcal{T}_c denote the minimal set of position terms satisfying the following conditions.

- for every $t_1 \sim t_2$ or $t_1 \not\sim t_2$ occurring in c , $t_1, t_2 \in \mathcal{T}_c$,
- for every $t \in \mathcal{T}_c$ and $t' \in \mathcal{C}_{rgd}[\Sigma^\pm]$ such that $t' \leq t$, we have $t', t[t' \setminus cur] \in \mathcal{T}_c$.

Similar to the construction of a NRAGs from NRRAs in the proof of Theorem 3, we can define concept of profiles with respect to \mathcal{T}_c . More specifically, a profile is a triple (S, χ, \sim) such that $S \subseteq \mathcal{T}_c$, \sim is an equivalence relation on S , and

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) (b_0, T_0, b'_0, s_0) (b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}) .$$

In addition, some consistency conditions can be defined such that c is satisfiable iff there is a consistent profile (S, χ, \sim) with $c \in T_0$ in χ .

Since the size of a profile is polynomial over that of c , a profile (S, χ, \sim) can be guessed and the consistency condition as well as $c \in T_0$ can be checked in polynomial time. Therefore, the satisfiability of rigid data constraints is in NP. \square

B. PROOFS IN SECTION 3.2

Proposition 2 *NRRAs and NRRAs are expressively incomparable.*

PROOF. The data language “there are two distinct positions with the same data value” is definable in NRAs, but not in NRRAs.

On the other hand, the data language “the sequence of word symbols belongs to ab^*a and the last data value does not occur elsewhere” is definable in NRRAs, but not in NRAs. \square

Proposition 3 *The class of languages definable by NRRAs are not closed under letter projections.*

PROOF. Let $\Sigma = \{(a, 0), (a, 1)\}$, $\Gamma = \{a\}$, and prj be a letter projection from Σ to Γ such that $prj((a, 0)) = prj((a, 1)) = a$.

Let L be the data language “there are *exactly* two distinct positions labeled by $(a, 1)$ and the data values before these two positions are the same”. Then $prj(L)$ is the data language “there are two distinct positions with the same data value”.

It is easy to see that L can be defined by a NRRA \mathcal{A} . On the other hand, $prj(L)$ is not definable by a NRRA.

We would like to remark that over the alphabet $\{a\}$, the position terms $suc_{\mathcal{A}}(t)$ (resp. $pred_{\mathcal{A}}(t)$) in $prj(\mathcal{A})$ are equal to $suc(t)$ (resp. $pred(t)$). Therefore, $prj(\mathcal{A})$ does not define $prj(L)$. \square

Proposition 4 *Suppose \mathcal{A} is a NRRA over Σ^\pm and prj is a letter projection from Σ^\pm to Γ . If \mathcal{A} is position-invariant under prj , then $\mathcal{L}(prj(\mathcal{A})) = prj(\mathcal{L}(\mathcal{A}))$.*

PROOF. Suppose \mathcal{A} is a NRRA over Σ^\pm and prj is a letter projection from Σ^\pm to Γ such that \mathcal{A} is position-invariant under prj .

For every position term $t \in \mathcal{T}_{\mathcal{A}}$, define $prj(t)$ as the position term obtained from t by replacing every occurrence of $suc_{\mathcal{A}}$ with $suc_{prj(\mathcal{A})}$. In addition, for every $c \in \mathcal{C}_{\mathcal{A}}$, define $prj(c)$ as the rigid data constraint obtained from c by replacing every position term t with $prj(t)$.

We first prove the following claim.

Claim. Let $\alpha = d_0 a_1 d_1 \dots a_n d_n$ be a data path, $c \in \mathcal{C}_{\mathcal{A}}$ and $i : 0 \leq i \leq n$. Then $(\alpha, 2i) \models c$ iff $(prj(\alpha), 2i) \models prj(c)$.

PROOF. It is sufficient to prove that for every $t \in \mathcal{T}_{\mathcal{A}}$ and $i : 0 \leq i \leq n$, $t_\alpha[2i] = (prj(t))_{prj(\alpha)}[2i]$.

This result can be proved by an induction on the structure of the position terms. In the following, we take $t = suc_{\mathcal{A}}(cur)$ as an example to illustrate the proof.

Suppose $(suc_{\mathcal{A}}(cur))_\alpha[2i] = 2j$ for some $j : i < j$. Then the first occurrence of word symbols from \mathcal{A} in α after the position $2i$ is in the position $2j$. It follows that the first occurrence of word symbols from $prj(\mathcal{A})$ in $prj(\alpha)$ after the

position $2i$ is in the position $2j$. Otherwise, there is $j' : i < j' < j$ such that a word symbol from $prj(A)$ occurs in the position $2j'$ of $prj(\alpha)$. From the fact that \mathcal{A} is position-invariant under prj , we know that $A = prj^{-1}(prj(A))$. Thus, a word symbol from A occurs in the position $2j' < 2j$ of α , a contradiction. Therefore, $(suc_{prj(A)}(cur))_{prj(\alpha)}[2i] = 2j$.

Suppose $(suc_{prj(A)}(cur))_{prj(\alpha)}[2i] = 2j$ for some $j : i < j$. Then the first occurrence of word symbols from $prj(A)$ in $prj(\alpha)$ after the position $2i$ is in the position $2j$. It follows that the first occurrence of word symbols from A in α after the position $2i$ is in the position $2j$. Otherwise, there is $j' : i < j' < j$ such that a word symbol from A occurs in the position $2j'$ of α . Thus, a word symbol from $prj(A)$ occurs in the position $2j' < 2j$ of $prj(\alpha)$, a contradiction. Therefore, $(suc_A(cur))_\alpha[2i] = 2j$. \square

$\mathcal{L}(prj(\mathcal{A})) \subseteq prj(\mathcal{L}(\mathcal{A}))$:

Suppose $\beta = d_0\gamma_1d_1 \dots \gamma_nd_n \in \mathcal{L}(prj(\mathcal{A}))$.

Then there is an accepting run of $prj(\mathcal{A})$ over β , say $\rho = q_0c_0q_1\gamma_1q_2c_1 \dots q_{2n-1}\gamma_nq_{2n}c_nq_{2n+1}$.

From the definition of $prj(\mathcal{A})$, we know that

- for every $i : 1 \leq i \leq n$, there is $a_i \in \Sigma^\pm$ such that $prj(a_i) = \gamma_i$ and $(q_{2i-1}, a_i, q_{2i}) \in \delta_w$,
- for every $i : 0 \leq i \leq n$, there is $c'_i \in \mathcal{C}_\mathcal{A}$ such that $(q_{2i}, c'_i, q_{2i+1}) \in \delta_d$ and $c_i = prj(c'_i)$.

Let $\alpha = d_0a_1d_1 \dots a_nd_n$. Then $\beta = prj(\alpha)$.

From the claim, we know that for every $i : 0 \leq i \leq n$, $(\alpha, 2i) \models c'_i$ iff $(prj(\alpha), 2i) \models c_i$.

Therefore, $q_0c'_0q_1a_1q_2 \dots q_{2n-1}a_nq_{2n}c'_nq_{2n+1}$ is an accepting run of \mathcal{A} over α . We conclude that $\alpha \in \mathcal{L}(\mathcal{A})$ and $\beta = prj(\alpha) \in prj(\mathcal{L}(\mathcal{A}))$.

$prj(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(prj(\mathcal{A}))$:

Let $\beta = d_0\gamma_1d_1 \dots \gamma_nd_n \in prj(\mathcal{L}(\mathcal{A}))$. Then there is $\alpha = d_0a_1d_1 \dots a_nd_n \in \mathcal{L}(\mathcal{A})$ such that $\beta = prj(\alpha)$. So there is an accepting run of \mathcal{A} over α , say $\rho = q_0c_0q_1a_1q_2c_1 \dots q_{2n-1}a_nq_{2n}c_nq_{2n+1}$.

From the claim, we know that for every $i : 0 \leq i \leq n$, $(\alpha, 2i) \models c_i$ iff $(prj(\alpha), 2i) \models prj(c_i)$. Therefore, $q_0prj(c_0)q_1prj(a_1)q_2prj(c_1) \dots q_{2n-1}prj(a_n)q_{2n}prj(c_n)q_{2n+1}$ is an accepting run of $prj(\mathcal{A})$ over $prj(\alpha) = \beta$. It follows that $\beta \in \mathcal{L}(prj(\mathcal{A}))$. \square

Proposition 5 *The nonemptiness problem of NRAGs is PSPACE-complete.*

PROOF. The upper bound:

Let $\mathcal{A} = (Q, k, \delta, I, F)$ be a NRAG. Then similar to NRAs ([10, 18]), a NFA $\mathcal{B} = (Q', \delta', I', F')$ can be constructed such that $\mathcal{L}(\mathcal{A})$ is nonempty iff $\mathcal{L}(\mathcal{B})$ is nonempty, and $|Q'|$ is polynomial over $|Q|$ and exponential over k . To decide the nonemptiness of \mathcal{B} , an accepting run of \mathcal{B} can be guessed nondeterministically in polynomial space. From Savitch's theorem, we know that the nonemptiness of \mathcal{A} can be decided in PSPACE.

The lower bound: Follows from that of NRAs. \square

Lemma 1. *Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRA over the alphabet Σ^\pm and $\alpha = d_0a_1d_1 \dots a_nd_n$ be a data path. Then for every run ρ of \mathcal{A} over α and every $i : 0 \leq i \leq n$, $Pos_\rho^f[\alpha, 2i] \cup Pos_\rho^p[\alpha, 2i] \subseteq \{t_\alpha[2i] \mid t \in \mathcal{T}_\mathcal{A}\}$.*

PROOF. Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRA, $\alpha = d_0a_1d_1 \dots a_nd_n$ be a data path, $\rho = q_0c_0q_1a_1q_2 \dots q_{2n-1}a_nq_{2n}c_nq_{2n+1}$ be a run of \mathcal{A} over α , and $i : 0 \leq i \leq n$.

Let $t_\alpha[2j] \in Pos_\rho^f[\alpha, 2i]$ such that $j \leq i$, t occurs in c_j , $t_\alpha[2j] \neq \perp$ and $t_\alpha[2j] > 2i$. Then there is $t' \in \mathcal{T}_\rho[\Sigma^\pm]$ such that $t' \leq t$, $t'_\alpha[2j] \leq 2i$, and for every $t'' : t' < t'' \leq t$, $t''_\alpha[2j] > 2i$. It follows that $suc(t') \leq t$ or $suc_A(t') \leq t$ for some $A \subseteq \Sigma^\pm$.

- If $suc(t') \leq t$, then $t'_\alpha[2j] = 2i$, since $(suc(t'))_\alpha[2j] = t'_\alpha[2j] + 1 > 2i$ and $t'_\alpha[2j] \leq 2i$. Thus, $t_\alpha[2j] = (t[t' \setminus cur])_\alpha[t'_\alpha[2j]] = (t[t' \setminus cur])_\alpha[2i]$.
- If $suc_A(t') \leq t$, then $t'_\alpha[2j] \leq 2i$ and $(suc_A(t'))_\alpha[2j] > 2i$. It follows that $a_{((suc_A(t'))_\alpha[2j])/2} \in A$, and for every $j' : i < j' < ((suc_A(t'))_\alpha[2j])/2$, $a_{j'} \notin A$. From this, it is deduced that $(suc_A(cur))_\alpha[2i] = (suc_A(t'))_\alpha[2j] = (suc_A(cur))_\alpha[t'_\alpha[2j]]$. Therefore, $t'_\alpha[2j] = 2i$, and $t_\alpha[2j] = (t[t' \setminus cur])_\alpha[t'_\alpha[2j]] = (t[t' \setminus cur])_\alpha[2i]$.

From the above argument, it follows that $Pos_\rho^f[\alpha, 2i] \subseteq \{t_\alpha[2i] \mid t \in \mathcal{T}_\mathcal{A}\}$. Similarly, we can show that $Pos_\rho^p[\alpha, 2i] \subseteq \{t_\alpha[2i] \mid t \in \mathcal{T}_\mathcal{A}\}$. \square

Theorem 3. *From a NRRA $\mathcal{A} = (Q, \delta, I, F)$, an equivalent NRAG $\mathcal{B} = (Q', k, \delta', I', F')$ can be constructed such that $|Q'|$ is polynomial over $|Q|$ and exponential over $|\mathcal{T}_\mathcal{A}|$ and k is polynomial over $|\mathcal{T}_\mathcal{A}|$.*

PROOF. Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRRA. In the following, we will construct a NRAG \mathcal{B} to simulate \mathcal{A} .

We first give an intuitive description of the construction. Let ρ be a run of \mathcal{A} over a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$. From Lemma 1, we know that for every $i : 0 \leq i \leq n$, $Pos_\rho^f[\alpha, 2i] \cup Pos_\rho^p[\alpha, 2i]$ contains only a bounded number of positions. It follows that only a bounded number of registers are needed to store them in the position $2i$. Therefore, \mathcal{B} can simulate ρ as follows: In the position $2i$,

- \mathcal{B} records in its registers the data values in the positions belonging to $Pos_\rho^p[\alpha, 2i]$.
- \mathcal{B} guesses in its registers the data values in the positions belonging to $Pos_\rho^f[\alpha, 2i]$.
- \mathcal{B} records the order for the positions in $Pos_\rho^p[\alpha, 2i]$ and $Pos_\rho^f[\alpha, 2i]$.

We introduce some additional notations.

Let $\alpha = d_0 a_1 d_1 \dots a_n d_n$ be a data path and $i : 0 \leq i \leq n$. The *profile* of the position $2i$ in α , denoted by $prof_\alpha(2i)$, is defined as a triple (S, χ, \sim) , where

- $S = \{t \in \mathcal{T}_\alpha \mid t_\alpha[2i] \neq \perp\}$,
- χ is a sequence

$$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) (b_0, T_0, b'_0, s_0) (b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

where

- for every $j : -m_1 \leq j \leq m_2$, $T_j \subseteq S$ and $T_j \neq \emptyset$,
- the collection $T_{-m_1}, \dots, T_0, \dots, T_{m_2}$ forms a partition of S ,
- for every $t, t' \in S$, if $t \in T_{j_1}$ and $t' \in T_{j_2}$, then $j_1 \leq j_2$ iff $t_\alpha[2i] \leq t'_\alpha[2i]$ (in particular, $j_1 = j_2$ iff $t_\alpha[2i] = t'_\alpha[2i]$),
- $cur \in T_0$,
- for every $j : -m_1 < j \leq m_2$, $b_j = a_{(t_\alpha[2i])/2}$ for some $t \in T_j$, and $b_{-m_1} = a_{(t_\alpha[2i])/2}$ if $t_\alpha[2i] > 0$ for some $t \in T_{-m_1}$, otherwise, $b_{-m_1} = \perp$,
- for every $j : -m_1 \leq j < m_2$, $b'_j = a_{(t_\alpha[2i])/2+1}$ for some $t \in T_j$, and $b'_{m_2} = a_{(t_\alpha[2i])/2+1}$ if $t_\alpha[2i] < 2n$ for some $t \in T_{m_2}$, otherwise, $b'_{m_2} = \perp$,
- $s_{m_2} = \perp$, and for every $j : -m_1 \leq j < m_2$, if $t'_\alpha(2i) = t_\alpha(2i) + 1$ for some $t \in T_j$ and $t' \in T_{j+1}$, then $s_j = 1$, otherwise, $s_j = 0$.
- \sim is an equivalence relation over S defined as follows: Let $t, t' \in S$, then $t \sim t'$ iff $d_{t_\alpha[2i]} = d_{t'_\alpha[2i]}$.

Let Σ_{prof} denote the set of all triples (S, χ, \sim) such that

- $S \subseteq \mathcal{T}_\alpha$,
- χ is a sequence

$$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) (b_0, T_0, b'_0, s_0) (b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

such that

- for every $j : -m_1 \leq j \leq m_2$, $T_j \neq \emptyset$,
- $cur \in T_0$,
- T_{-m_1}, \dots, T_{m_2} is a partition of S ,
- $b_{-m_1} \in \Sigma^\pm \cup \{\perp\}$, and for every $j : -m_1 < j \leq m_2$, $b_j \in \Sigma^\pm$,
- $b'_{m_2} \in \Sigma^\pm \cup \{\perp\}$, and for every $j : -m_1 \leq j < m_2$, $b'_j \in \Sigma^\pm$,
- $s_{m_2} = \perp$, and for every $j : -m_1 \leq j < m_2$, $s_j \in \{0, 1\}$,
- \sim is an equivalence relation over S such that for every $t, t' \in \mathcal{T}_\alpha$, if $t, t' \in T_j$ for some j , then $t \sim t'$.

Note that for $(S, \chi, \sim) \in \Sigma_{prof}$, there may be no data paths α and a position in α such that the profile of the position in α is (S, χ, \sim) . Nevertheless, we are able to define a consistency condition on the elements from Σ_{prof} so that a consistent element from Σ_{prof} indeed corresponds to the profile of a position in some data path. Moreover, for two consistent elements from Σ_{prof} , say $(S_1, \chi_1, \sim_1), (S_2, \chi_2, \sim_2)$, and $a \in \Sigma^\pm$, we are able to define a syntactic successor relation $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$, which mimics the changes from $prof_\alpha(2i)$ to $prof_\alpha(2(i+1))$ by reading a word symbol a in the position $2i+1$ of a data path.

For $A \subseteq \Sigma^\pm$, a sequence

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) (b_0, T_0, b'_0, s_0) (b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}) ,$$

and $j : -m_1 \leq j \leq m_2$, A is said to occur after (resp. before) T_j in χ if $b_{j'} \in A$ for some $j' : j < j' \leq m_2$ or $b'_{j'} \in A$ for some $j' : j \leq j' < -m_1$ (resp. $b'_{j'} \in A$ for some $j' : -m_1 \leq j' < j$ or $b_{j'} \in A$ for some $j' : -m_1 \leq j' \leq j$).

Let $(S, \chi, \sim) \in \Sigma_{prof}$ and

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}) .$$

Then (S, χ, \sim) is said to be *consistent* if χ satisfies the following conditions.

- For every $suc(t) \in \mathcal{T}_A$ (resp. $pred(t) \in \mathcal{T}_A$), and every $j : -m_1 \leq j < m_2$ (resp. $j : -m_1 < j \leq m_2$), $t \in T_j$ iff $suc(t) \in T_{j+1}$ (resp. $t \in T_j$ iff $pred(t) \in T_{j-1}$).
- For every $suc(t) \in \mathcal{T}_A$ and every $j : -m_1 \leq j < m_2$, if $t \in T_j$ and $suc(t) \in T_{j+1}$, then $b'_j = b_{j+1}$ and $s_j = 1$.
- For every $pred(t) \in \mathcal{T}_A$ and every $j : -m_1 < j \leq m_2$, if $t \in T_j$ and $pred(t) \in T_{j-1}$, then $b'_{j-1} = b_j$ and $s_{j-1} = 1$.
- For every $suc_A(t) \in \mathcal{T}_A$, if $suc_A(t) \in T_j$ for some $j : -m_1 < j \leq m_2$, then $b_j \in A$.
- For every $pred_A(t) \in \mathcal{T}_A$, if $pred_A(t) \in T_j$ for some $j : -m_1 \leq j < m_2$, then $b'_j \in A$.
- For every $suc_A(t) \in \mathcal{T}_A$ (resp. $pred_A(t) \in \mathcal{T}_A$), if $suc_A(t) \in T_j$ (resp. $pred_A(t) \in T_j$), then $t \in T_{j'}$ for some $j' : j' < j$ (resp. $j' : j' > j$).
- For every $suc_A(t) \in \mathcal{T}_A$, if $t \in T_{j_1}$ and $suc_A(t) \in T_{j_2}$ ($j_1 < j_2$), then for every $j_3 : j_1 < j_3 < j_2$, $b_{j_3} \notin A$, and for every $j_3 : j_1 \leq j_3 < j_2 - 1$, $b'_{j_3} \notin A$; in addition, $b'_{j_2-1} \in A$ implies $s_{j_2-1} = 1$.
- For every $pred_A(t) \in \mathcal{T}_A$, if $pred_A(t) \in T_{j_1}$ and $t \in T_{j_2}$ ($j_1 < j_2$), then for every $j_3 : j_1 < j_3 < j_2$, $b'_{j_3} \notin A$, and for every $j_3 : j_1 + 1 < j_3 \leq j_2$, $b_{j_3} \notin A$; in addition, $b_{j_1+1} \in A$ implies $s_{j_1} = 1$.
- For every $t, suc_A(t) \in \mathcal{T}_A$, if $t \in T_j$ for $j : -m_1 \leq j \leq m_2$, and A occurs after T_j , then $suc_A(t) \in T_{j'}$ for some $j' : j' > j$.
- For every $t, pred_A(t) \in \mathcal{T}_A$, if $t \in T_j$ for $j : -m_1 \leq j \leq m_2$, and A occurs before T_j , then $pred_A(t) \in T_{j'}$ for some $j' : j' < j$.

Claim. Suppose $(S, \chi, \sim) \in \Sigma_{prof}$. Then (S, χ, \sim) is consistent iff there is a data path α and a position $2i$ in α such that $prof_\alpha(2i) = (S, \chi, \sim)$.

PROOF. The “if” direction is trivial.

The “only if” direction:

Suppose (S, χ, \sim) is consistent. Let

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}) .$$

For each T_j , we assign a data value d_j , in a way that respects the equivalence relation \sim , that is, if $t \in T_j$, $t' \in T_{j'}$, and $t \sim t'$, then $d_j = d_{j'}$. In addition, let d be a data value different from all these d_j 's.

For every $j : -m_1 \leq j \leq m_2$, we define a data path α_j as follows.

- For $j = -m_1$,
 - if $s_{-m_1} = 1$ and $b_{-m_1} \neq \perp$, then let $\alpha_{-m_1} = db_{-m_1}d_{-m_1}$,
 - if $s_{-m_1} = 1$ and $b_{-m_1} = \perp$, then let $\alpha_{-m_1} = d_{-m_1}$,
 - if $s_{-m_1} = 0$ and $b_{-m_1} \neq \perp$, then let $\alpha_{-m_1} = db_{-m_1}d_{-m_1}b'_{-m_1}d$,
 - if $s_{-m_1} = 0$ and $b_{-m_1} = \perp$, then let $\alpha_{-m_1} = d_{-m_1}b'_{-m_1}d$.
- For $j : -m_1 < j < m_2$,
 - if $s_{j-1} = 1$ and $s_j = 1$, then let $\alpha_j = d_{j-1}b_jd_j$,
 - if $s_{j-1} = 1$ and $s_j = 0$, then let $\alpha_j = d_{j-1}b_jd_jb'_jd$,
 - if $s_{j-1} = 0$ and $s_j = 1$, then let $\alpha_j = db_jd_j$,
 - if $s_{j-1} = 0$ and $s_j = 0$, then let $\alpha_j = db_jd_jb'_jd$.
- For $j = m_2$,
 - if $s_{m_2-1} = 1$ and $b'_{m_2} \neq \perp$, then let $\alpha_{m_2} = d_{m_2-1}b_{m_2}d_{m_2}b'_{m_2}d$,
 - if $s_{m_2-1} = 1$ and $b'_{m_2} = \perp$, then let $\alpha_{m_2} = d_{m_2-1}b_{m_2}d_{m_2}$,
 - if $s_{m_2-1} = 0$ and $b'_{m_2} \neq \perp$, then let $\alpha_{m_2} = db_{m_2}d_{m_2}b'_{m_2}d$,
 - if $s_{m_2-1} = 0$ and $b'_{m_2} = \perp$, then let $\alpha_{m_2} = db_{m_2}d_{m_2}$.

Consider the data path $\alpha = \alpha_{-m_1} \cdot \alpha_{-m_1+1} \cdot \dots \cdot \alpha_{-1} \cdot \alpha_0 \cdot \alpha_1 \cdot \dots \cdot \alpha_{m_2}$.

For every $j : -m_1 \leq j \leq m_2$, let the position of α corresponding to the data value d_j be $2i_j$.

From the construction of α from (S, χ, \sim) , by an induction on the structure of position terms, we can prove the following result.

For every $t \in \mathcal{T}_A$ and every $j : -m_1 \leq j \leq m_2$, $t \in T_j$ iff $t_\alpha[2i_0] = 2i_j$. (*)

Let us take $t = \text{suc}_A(\text{cur})$ as an example to illustrate the proof.

Suppose $\text{suc}_A(\text{cur}) \in T_j$, then $b_j \in A$ and for every $j' : 0 < j' < j$, $b_{j'} \notin A$, and for every $j' : 0 \leq j' < j-1$, $b'_{j'} \notin A$; in addition, $b'_{j-1} \in A$ implies $s_{j-1} = 1$. From this, we deduce that $(\text{suc}_A(\text{cur}))_\alpha[2i_0] = 2i_j$, since all the word symbols located after the position $2i_0$ and before the position $2(i_j - 1)$ in α do not belong to A .

On the other hand, suppose $(\text{suc}_A(\text{cur}))_\alpha[2i_0] = 2i_j$, then the word symbol immediately before the position $2i_j$, that is, b_j , belongs to A , and all the word symbols located after the position $2i_0$ and before the position $2(i_j - 1)$ in α do not belong to A . From the construction of α , it follows that $b_j \in A$ and for every $j' : 0 < j' < j$, $b_{j'} \notin A$, and for every $j' : 0 \leq j' < j-1$, $b'_{j'} \notin A$; in addition, $b'_{j-1} \in A$ implies $s_{j-1} = 1$ and $b'_{j-1} = b_j$. From this, we conclude that $\text{suc}_A(\text{cur}) \in T_j$.

From the result (*), we conclude that $\text{prof}_\alpha[2i_0] = (S, \chi, \sim)$. \square

Let Prof_A denote the set of elements of Σ_{prof} that are consistent. Suppose $(S, \chi, \sim) \in \text{Prof}_A$,

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) (b_0, T_0, b'_0, s_0) (b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}),$$

and $c \in \mathcal{C}_A$. Then the satisfaction of c over (S, χ, \sim) , denoted by $(S, \chi, \sim) \models c$, can be defined by interpreting c over (S, χ, \sim) in a natural way. For instance, if $c = t_1 \sim t_2$, then $(S, \chi, \sim) \models c$ if $t_1, t_2 \in S$ and $t_1 \sim t_2$.

Suppose $a \in \Sigma^\pm$, $(S_1, \chi_1, \sim_1), (S_2, \chi_2, \sim_2) \in \text{Prof}_A$, for $i = 1, 2$,

$$\chi_i = (b_{i,-m_{i,1}}, T_{i,-m_{i,1}}, b'_{i,-m_{i,1}}, s_{i,-m_{i,1}}) \dots (b_{i,-1}, T_{i,-1}, b'_{i,-1}, s_{i,-1}) \\ (b_{i,0}, T_{i,0}, b'_{i,0}, s_{i,0}) (b_{i,1}, T_{i,1}, b'_{i,1}, s_{i,1}) \dots (b_{i,m_{i,2}}, T_{i,m_{i,2}}, b'_{i,m_{i,2}}, s_{i,m_{i,2}}).$$

In the following, we will define the concept that (S_2, χ_2, \sim_2) is a *successor* of (S_1, χ_1, \sim_1) with respect to a , denoted by $(S_2, \chi_2, \sim_2) \xrightarrow{a} (S_1, \chi_1, \sim_1)$.

For every $j : -m_{1,1} \leq j \leq m_{1,2}$, construct $T'_{1,j} \subseteq \mathcal{T}_A$ from $T_{1,j}$ as follows.

- For every $t \in T_{1,j}$ such that $\text{suc}(\text{cur}) \leq t$, let $t[\text{suc}(\text{cur})\backslash\text{cur}] \in T'_{1,j}$.
- For every $t \in T_{1,j}$ and $A \subseteq \Sigma^\pm$ such that $a \in A$ and $\text{suc}_A(\text{cur}) \leq t$, let $t[\text{suc}_A(\text{cur})\backslash\text{cur}] \in T'_{1,j}$.
- For every $t \in T_{1,j}$ such that $t[\text{cur}\backslash\text{pred}(\text{cur})] \in \mathcal{T}_A$, let $t[\text{cur}\backslash\text{pred}(\text{cur})] \in T'_{1,j}$.
- For every $t \in T_{1,j}$ and $A \subseteq \Sigma^\pm$ such that $a \in A$ and $t[\text{cur}\backslash\text{pred}_A(\text{cur})] \in \mathcal{T}_A$, let $t[\text{cur}\backslash\text{pred}_A(\text{cur})] \in T'_{1,j}$.
- For every $t \in T_{1,j}$ such that $a \notin A$, $\text{suc}_A(\text{cur}) \leq t$ or $\text{pred}_A(\text{cur}) \leq t$, let $t \in T'_{1,j}$.
- If $s_{1,0} = 1$, let $\text{cur} \in T'_{1,1}$.

Note that $T'_{1,j}$'s defined above may be empty for some $j : -m_1 \leq j \leq m_2$.

$(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$ if the following conditions hold.

- $b'_{1,0} = a$.
- Let $j_1 j_2 \dots j_\ell : -m_{1,1} \leq j_1 < j_2 < \dots < j_\ell \leq 0$ be the sequence of the *non-positive* indices such that for every $r : 1 \leq r \leq \ell$, $T'_{1,j_r} \neq \emptyset$ (and all the other $T'_{1,j}$'s for non-positive j 's are empty). Then the sequence

$$(b_{2,-m_{2,1}}, T_{2,-m_{2,1}}, b'_{2,-m_{2,1}}, s_{2,-m_{2,1}}) \dots (b_{2,-1}, T_{2,-1}, b'_{2,-1}, s_{2,-1})$$

is equal to the sequence $(b_{1,j_1}, T'_{1,j_1}, b'_{1,j_1}, s_{1,j_1}) \dots (b_{1,j_\ell}, T'_{1,j_\ell}, b'_{1,j_\ell}, s_{1,j_\ell})$.

- There is a partial mapping f from $\{1, \dots, m_{1,2}\}$ to $\{0, \dots, m_{2,2}\}$ such that
 - for every $j : 1 \leq j \leq m_{1,2}$, $f(j)$ is undefined iff $T'_{1,j} = \emptyset$,
 - f is increasing, that is, if $j_1 < j_2$ and $f(j_1), f(j_2)$ are defined, then $f(j_1) < f(j_2)$,
 - for every $j : 1 \leq j \leq m_{1,2}$, if $f(j)$ is defined, then $T'_{1,j} \subseteq T_{2,f(j)}$, $b_{1,j} = b_{2,f(j)}$ and $b'_{1,j} = b'_{2,f(j)}$,
 - if $f(1)$ is defined, then $f(1) = 0$ iff $s_{1,0} = 1$,
 - for every $j : 1 \leq j < m_{1,2}$, if $f(j), f(j+1)$ are both defined, then $s_{1,j} = 1$ implies $f(j+1) = f(j) + 1$ and $s_{2,f(j)} = 1$,

- for every $j_1, j_2 : j_1 < j_2$, if $f(j_1), f(j_2)$ are both defined, then for every $t \in T_{2,f(j_1)}, t' \in T_{2,f(j_2)}$, $t \sim_2 t'$ iff there are $t_1 \in T_{1,j_1}, t'_1 \in T_{1,j_2}$ such that $t_1 \sim_1 t'_1$.

Intuitively, (S_1, χ_1, \sim_1) is rotated one-position to the left to get the profile (S_2, χ_2, \sim_2) . The $T'_{1,j}$'s together with f above define the information that should be inherited during the rotation.

We are ready to construct the NRAG \mathcal{B} .

There are $2|\mathcal{T}_A| + 1$ registers in \mathcal{B} , that is,

$$r_1, \dots, r_{|\mathcal{T}_A|}, r_{|\mathcal{T}_A|+1}, \dots, r_{2|\mathcal{T}_A|}.$$

Over a data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$, \mathcal{B} does the following.

- In each position $2i$ ($0 \leq i \leq n$), \mathcal{B} guesses $\pi_i = (S_i, \chi_i, \sim_i) \in \text{Prof}_A$ (where π_i is supposed to be $\text{prof}_\alpha[2i]$). In addition,
 - if $i = 0$, then $\pi_0 = (S_0, \chi_0, \sim_0)$ is an initial profile, that is, for every $t \in \mathcal{T}_A$ such that $\text{pred}(\text{cur}) \leq t$ or $\text{pred}_A(\text{cur}) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_0$,
 - if $i = n$, then $\pi_n = (S_n, \chi_n, \sim_n)$ is a final profile, that is, for every $t \in \mathcal{T}_A$ such that $\text{suc}(\text{cur}) \leq t$ or $\text{suc}_A(\text{cur}) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_n$.
- For every $i : 0 \leq i \leq n$, if

$$\chi_i = \begin{pmatrix} (b_{i,-m_{i,1}}, T_{i,-m_{i,1}}, b'_{i,-m_{i,1}}, s_{i,-m_{i,1}}) \dots \\ (b_{i,-1}, T_{i,-1}, b'_{i,-1}, s_{i,-1}) (b_{i,0}, T_{i,0}, b'_{i,0}, s_{i,0}) \\ (b_{i,1}, T_{i,1}, b'_{i,1}, s_{i,1}) \dots \\ (b_{i,m_{i,2}}, T_{i,m_{i,2}}, b'_{i,m_{i,2}}, s_{i,m_{i,2}}) \end{pmatrix},$$

then after the position $2i$ is visited (that is, the reading head is in $2i + 1$), for each $j : -m_{i,1} \leq j \leq m_{i,2}$, \mathcal{B} stores in the register $r_{j+|\mathcal{T}_A|}$ the data value corresponding to $T_{i,j}$. In particular, \mathcal{B} stores the data value d_i in $r_{|\mathcal{T}_A|}$.

- Over each pair of positions $2i$ and $2(i + 1)$ (where $0 \leq i < n$), \mathcal{B} checks that $\pi_i \xrightarrow{a_{i+1}} \pi_{i+1}$. To do this, \mathcal{B} copies (by guessing) data values between registers and guesses some data values for a few registers.
- At the same time, \mathcal{B} simulates the run of \mathcal{A} as follows.
 - If \mathcal{A} makes a transition (q, a_i, q') over a_i , then \mathcal{B} checks that $b'_{i-1,0} = a_i$ and changes the state from q to q' .
 - If \mathcal{A} makes a transition (q, c, q') over d_i , then \mathcal{B} checks that π_i satisfies c , verifies that d_i is equal to the data value stored in $r_{j+|\mathcal{T}_A|}$ for each $j : -m_{i,1} \leq j \leq m_{i,2}$ such that there is $t \in T_j$ satisfying $\text{cur} \sim_i t$ (in particular, d_i should be equal to the data value in $r_{|\mathcal{T}_A|}$), and changes the state from q to q' .
 - \mathcal{B} accepts if \mathcal{A} accepts and a final profile is reached.

From the above construction, we know that in its states, \mathcal{B} should record the states of \mathcal{A} and the guessed profiles. Therefore, the number of states of \mathcal{B} is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. \square

Proposition 6 *The nonemptiness of NRRAs and DRRAs is PSPACE-complete.*

PROOF. The upper bound:

From Theorem 3, given a NRRA $\mathcal{A} = (Q, \delta, I, F)$, an equivalent NRAG $\mathcal{B} = (Q', k, \delta', I', F')$ can be constructed such that $\mathcal{L}(\mathcal{A})$ is nonempty iff $\mathcal{L}(\mathcal{B})$ is nonempty. Moreover, \mathcal{B} satisfies that $|Q'|$ is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$, and k is polynomial over $|\mathcal{T}_A|$.

From the proof of Proposition 5, we know that a NFA \mathcal{B}' can be constructed from \mathcal{B} such that $\mathcal{L}(\mathcal{B}')$ is nonempty iff $\mathcal{L}(\mathcal{B})$ is nonempty. Since the number of states of \mathcal{B}' is polynomial over $|Q'|$ and exponential over k , it follows that the number of states of \mathcal{B}' is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. To decide the nonemptiness of \mathcal{A} , an accepting run of \mathcal{B}' can be guessed nondeterministically in polynomial space. The PSPACE upper bound then follows from Savitch's theorem.

The lower bound:

The reduction from the membership problem of polynomial space Turing machines to the nonemptiness problem of NRAs or DRAs ([10]) can be adapted to a reduction to the nonemptiness problem of NRRAs or DRRAs. \square

Proposition 7 *For every NRRA \mathcal{A} , there is an equivalent DRRA of exponential size.*

PROOF. Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRRA. We construct a DRRA $\mathcal{A}' = (Q', \delta', q'_0, F')$ as follows:

- $Q' = Q'_w \cup Q'_d$, where $Q'_w = 2^{Q_w}$, $Q'_d = 2^{Q_d}$,
- $q'_0 = I, F' = \{S \in Q'_w \mid S \cap F \neq \emptyset\}$,

- $\delta' = \delta'_w \cup \delta'_d$ is defined as follows:
 - $\delta'_w = \{(S, a, S') \mid S \in Q'_w, S' \in Q'_d, S' = \{q' \mid \exists q \in S. (q, a, q') \in \delta_w\}\}$,
 - δ'_d is defined as follows:
 For every $S \in Q'_d$, let C denote the set of rigid data constraints occurring in the tuples $(q, c, q') \in \delta_d$ such that $q \in S$. Then δ'_d contains all tuples (S, c', S') such that there exists $C' \subseteq C$ satisfying that $c' = \bigwedge_{c \in C'} c \wedge \bigwedge_{c \in C \setminus C'} \bar{c}$, and $S' = \{q' \mid \exists q \in S, c \in C'. (q, c, q') \in \delta_d\}$.

Note that the transitions (S, c', S') may be non-applicable if c' is unsatisfiable.

If $(S, c'_1, S'_1), (S, c'_2, S'_2) \in \delta'_d$ such that $S'_1 \neq S'_2$, then there are C'_1, C'_2 such that $C'_1 \neq C'_2$, $c'_1 = \bigwedge_{c \in C'_1} c \wedge \bigwedge_{c \in C \setminus C'_1} \bar{c}$ and $c'_2 = \bigwedge_{c \in C'_2} c \wedge \bigwedge_{c \in C \setminus C'_2} \bar{c}$. It is easy to observe that if $C'_1 \neq C'_2$, then $c'_1 \wedge c'_2$ is unsatisfiable. Therefore, \mathcal{A}' is a DRRA. \square

Corollary 2 *The language inclusion problem for NRRAs is PSPACE-complete.*

PROOF. The upper bound:

Let $\mathcal{A} = (Q_1, \delta_1, I_1, F_1)$ and $\mathcal{B} = (Q_2, \delta_2, I_2, F_2)$ be two NRRAs. To decide whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$, we use the following procedure.

- Determinize and complement \mathcal{B} , let \mathcal{C} be the resulting DRRA.
- Construct the product of \mathcal{A} and \mathcal{B} , say \mathcal{C}' , that defines $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{C})$. Check whether $\mathcal{L}(\mathcal{C}') \neq \emptyset$.

From the proof of Proposition 7, we know that the size of \mathcal{C} is exponential over $|Q_2|$. Thus, the size of \mathcal{C}' is polynomial over $|Q_1|$ and exponential over $|Q_2|$. The set of position terms of \mathcal{C}' is the union of $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}_{\mathcal{B}}$.

From the proof of Proposition 6, it follows that the nonemptiness of \mathcal{C}' can be reduced to that of a NFA of size polynomial over $|Q_1|$, exponential over $|Q_2|$, and exponential over $|\mathcal{T}_{\mathcal{A}}|, |\mathcal{T}_{\mathcal{B}}|$.

From Savitch's theorem, we conclude that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ can be decided in PSPACE.

The lower bound:

The language inclusion of NFAs is already PSPACE-hard. \square

C. PROOFS IN SECTION 3.3

Proposition 8. *For every 2NRRRA, there is an equivalent NRRRA of exponential size.*

PROOF. The proof is an adaptation of Shepherdson's method [20] to construct an equivalent NFA from a two-way NFA.

Let $\mathcal{A} = (Q, \vdash, \dashv, \delta, I, F)$ be a 2NRRRA. We construct a NRRRA $\mathcal{A}' = (Q', \delta', I', F')$ as follows.

- $Q' = Q'_d \cup Q'_w$, where
 - Q'_d is the set of all tuples $(C, f) \in 2^{\mathcal{C}_{\mathcal{A}}} \times (Q \cup \{\perp\})^{Q_d \cup \{\cdot\}}$ such that $\bigwedge_{c \in C} c$ is satisfiable, $f(\cdot) \in Q_d$, and for every $q \in Q_d$, $f(q) \in Q_w \cup \{\perp\}$,
 - Q'_w is the set of all tuples $(a, f) \in (\Sigma^{\pm} \cup \{\dashv\}) \times (Q \cup \{\perp\})^{Q_w \cup \{\cdot\}}$ such that $f(\cdot) \in Q_w$, and for every $q \in Q_w$, $f(q) \in Q_d \cup \{\perp\}$.
- I' is the set of $(C, f) \in Q'_d$ satisfying the following conditions,
 - $f(\cdot) = q$ such that $(q', \vdash, q, +1) \in \delta_w$ for some $q' \in I$,
 - if $f(q) = q'$, then there exist $q_0, q_1, \dots, q_k \in Q_d$ and $p_1, \dots, p_k \in Q_w$ such that
 - * $q_0 = q$,
 - * for every $i : 0 \leq i < k$, there is $c_i \in C$ such that $(q_i, c_i, p_{i+1}, -1) \in \delta_d$, and for every $i : 1 \leq i \leq k$, $(p_i, \vdash, q_i, +1) \in \delta_w$,
 - * there is $c \in C$ such that $(q_k, c, q', +1) \in \delta_d$.
- δ' are defined as follows.
 - Let $(a, f) \in Q'_w, (C, f') \in Q'_d$. Then $((a, f), a, (C, f')) \in \delta'_w$ if for every pair (q, q') such that $q \in Q_d, q' \in Q_w$ and $f'(q) = q'$, the following condition holds.

There exist $q_0, q_1, \dots, q_k \in Q_d, p_1, \dots, p_k \in Q_w$ such that $q_0 = q$, and the following conditions hold,

 1. for every $0 \leq i < k$, there exists $c \in C$ such that $(q_i, c, p_{i+1}, -1) \in \delta_d$,
 2. for every $1 \leq i \leq k$, $f(p_i) = q_i$,

3. there is $c \in C$ such that $(q_k, c, q', +1) \in \delta$.
- Let $(C, f) \in Q'_d, (a, f') \in Q'_w$. Then $((C, f), \bigwedge_{c \in C} c, (a, f')) \in \delta'_d$ iff the following conditions hold.
 For every pair (q, q') such that $q, q' \in Q$ and $f'(q) = q'$, there exist $q_0, q_1, \dots, q_k \in Q_w$ and $p_1, \dots, p_k \in Q_d$ satisfying that $q_0 = q$, and
 1. for every $0 \leq i < k$, $(q_i, a, p_{i+1}, -1) \in \delta_w$,
 2. for every $1 \leq i \leq k$, $f(p_i) = q_i$,
 3. $(q_k, a, q', +1) \in \delta_w$.
- F' consists of all $(\neg, f) \in Q'_w$ such that
 - $f(\cdot) = q \in Q_w$,
 - for every $q' \in Q_w$, $f(q') = \perp$,
 - there is $(C, f') \in Q'_d$ such that $((C, f'), \bigwedge_{c \in C} c, f) \in \delta'_d$, there are $p_1, \dots, p_k \in Q_d$ and $q_0, \dots, q_k \in Q_w$ satisfying that $q_0 = q$, for every $i : 0 \leq i < k$, $(q_i, \neg, p_{i+1}, -1) \in \delta_w$, and for every $i : 0 \leq i \leq k$, $f'(p_i) = q_i$, and $q_k \in F$.

Now we prove the correctness of the construction, that is, for every data path $\alpha = d_0 a_1 d_1 \dots a_n d_n$, \mathcal{A} accepts α iff \mathcal{A}' accepts α .

“Only if direction”:

Suppose \mathcal{A} accepts α . Then there is an accepting run of \mathcal{A} over α , say $(q_0, i_0)\theta_0(q_1, i_1)\theta_1 \dots \theta_{m-1}(q_m, i_m)$, such that

- $q_0 \in I, q_m \in F$,
- $i_0 = 0, i_m = 2n + 2$,
- for every $j : 0 \leq j < m$, if i_j is even, then there is $dir \in \{+1, -1\}$ such that $(q_j, a_{i_j/2}, q_{j+1}, dir) \in \delta_w$ (where $a_0 = \vdash, a_{n+1} = \neg$), $\theta_j = a_{i_j/2}$, and $i_{j+1} = i_j + dir$,
- for every $j : 0 \leq j \leq m$, if i_j is odd, then there are $c \in \mathcal{C}_{rgd}$ and $dir \in \{+1, -1\}$ such that $(q_j, c, q_{j+1}, dir) \in \delta_d$, $(\alpha, i_j - 1) \models c$, $\theta_j = c$, and $i_{j+1} = i_j + dir$.

Without loss of generality, we assume that in the accepting run above, no states are repeated when the reading head moves to the same position, more precisely, the following condition holds.

For every $j_1, j_2 : 0 < j_1 < j_2 < m$ such that $i_{j_1} = i_{j_2}$, it holds that $q_{i_{j_1}} \neq q_{i_{j_2}}$. (*)

The above assumption is justified by the fact that if a state is repeated in the same position, then the subrun between the repetitions can be trimmed and the remaining part is still an accepting run.

For each $i : 1 \leq i \leq 2n + 2$, define f_i as follows.

1. For every $q \in Q$, if there are $j_1, j_2 : 0 \leq j_1 < j_2 < m$ such that

- $q_{j_1} = q, q_{j_2} = q'$,
- $i_{j_1} = i, i_{j_2} = i + 1$, and for every $j' : j_1 < j' < j_2$, $i_{j'} \leq i$,

then $f_i(q) = q'$, otherwise $f_i(q) = \perp$.

Note that the assumption (*) guarantees that for every i , there is at most one pair (j_1, j_2) satisfying the above condition. So f_i is well-defined.

2. $f_i(\cdot) = q$, where $q \in Q$ satisfies that there exists $j : 0 < j \leq m$ such that $i_j = i$, $q = q_{i_j}$, and for every $j' : 0 \leq j' < j$, $i_{j'} < i$.

For each $i : 0 \leq i \leq n$, define $C_i \subseteq \mathcal{C}_{\mathcal{A}}$ as the set of θ_j 's such that $i_j = 2i + 1$.

Then

$$(C_0, f_1) \left(\bigwedge_{c \in C_0} c \right) (a_1, f_2) a_1 (C_1, f_3) \dots (a_n, f_{2n}) a_n (C_n, f_{2n+1}) \left(\bigwedge_{c \in C_n} c \right) (\neg, f_{2n+2})$$

is an accepting run of \mathcal{A}' over α .

“If direction”:

Suppose $(C_0, f_0)c_0(a_1, f_1)a_1(C_1, f_2) \dots (a_n, f_{2n-1})a_n(C_n, f_{2n})c_n(\neg, f_{2n+1})$ is an accepting run of \mathcal{A}' over α . Since $f_{2n+1} \in F'$, there exist $q_0, q_1, \dots, q_k \in Q_w$ and $p_1, \dots, p_k \in Q_d$ such that

- $f_{2n+1}(\cdot) = q_0$,
- for every $i : 0 \leq i < k$, $(q_i, \neg, p_{i+1}, -1) \in \delta_w$, and for every $i : 1 \leq i \leq k$, $f_{2n}(p_i) = q_i$,

- $q_k \in F$.

From the fact that $f_{2n+1}(\cdot) = q_0 \neq \perp$, we deduce from the definition of δ' in \mathcal{A}' that $f_i(\cdot) \neq \perp$ for every $i : 0 \leq i \leq 2n$. It follows that for every $i : 1 \leq i \leq 2n+1$, $f_i(\cdot) = f_{i-1}(f_{i-1}(\cdot))$.

From the fact that $f_0(\cdot) \neq \perp$, there is $p_0 \in I$ such that $(p_0, \vdash, f_0(\cdot), +1) \in \delta_w$.

By induction on $i : 0 \leq i \leq 2n$, we can show that if $f_i(q) = q'$, then there is a subrun $\rho_{q,q'}^i$ from $(q, i+1)$ to $(q', i+2)$ of \mathcal{A} over $\vdash \alpha \dashv$.

Consider the composition of the following subruns,

$$(p_0, 0) \vdash (f_0(\cdot), 1), \rho_{f_0(\cdot), f_0(f_0(\cdot))}^1, \rho_{f_1(\cdot), f_1(f_1(\cdot))}^2, \dots, \rho_{f_{2n}(\cdot), f_{2n}(f_{2n}(\cdot))}^{2n}, (q_0, 2n+2) \dashv (p_1, 2n+1), \rho_{p_1, q_1}^{2n}, \\ (q_1, 2n+2) \dashv (p_2, 2n+1), \rho_{p_2, q_2}^{2n}, \dots, (q_{k-1}, 2n+2) \dashv (p_k, 2n+1), \rho_{p_k, q_k}^{2n}.$$

Let ρ denote this composition. Then ρ is an accepting run of \mathcal{A} over α . \square

D. PROOFS IN SECTION 4

Proposition 10 *The evaluation for 2RRDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.*

PROOF. The upper bound:

We use the idea to prove the PSPACE upper bound for 2RDPQs in [18].

Let $\mathcal{G} = (V, E, \eta)$ be a data graph, $\xi = (x, L, y)$ be a 2RRDPQ, and $(v_1, v_2) \in V \times V$. Suppose L is given by a NRRRA $\mathcal{A} = (Q, \delta, I, F)$ over Σ^\pm .

Let D be the set of data values occurring in \mathcal{G} . Then \mathcal{G} plus (v_1, v_2) can be seen as a NFA $\mathcal{A}_{\mathcal{G}, (v_1, v_2)} = (Q', \delta', I', F')$ with initial state $(v_1)_s$ and final state $(v_2)_t$ over the alphabet $\Sigma^\pm \cup D$ as follows.

- $Q' = \{v_s, v_t \mid v \in V\}$,
- $\delta' = \{(v_t, a, v'_s), (v'_t, a^-, v_s) \mid (v, a, v') \in E\} \cup \{(v_s, d, v_t) \mid v \in V, \eta(v) = d\}$,
- $I' = \{v_1\}$, $F' = \{v_2\}$.

From the proof of Theorem 3, we know that from \mathcal{A} , an equivalent NRAG $\mathcal{B} = (Q', k, \delta', I', F')$ can be constructed such that $|Q'|$ is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$, and k is polynomial over $|\mathcal{T}_A|$.

When restricted to the data paths where all data values are from D , the NRAG \mathcal{B} can be seen as a NFA \mathcal{B}' over the alphabet $\Sigma^\pm \cup D$ with the state space $Q' \times D^{[k]}$. It follows that the size of the state space of \mathcal{B}' is exponential over the size of \mathcal{A} and polynomial over the size of D .

To decide whether $(v_1, v_2) \in \xi(\mathcal{G})$, it is sufficient to check whether $\mathcal{L}(\mathcal{A}_{\mathcal{G}, (v_1, v_2)} \cap \mathcal{B}') \neq \emptyset$. Since an accepting run of $\mathcal{A}_{\mathcal{G}, (v_1, v_2)} \cap \mathcal{B}'$ can be guessed in polynomial space, from Savitch's theorem, we conclude that the evaluation problem of NRRAs is in PSPACE.

If the size of \mathcal{A} is bounded by a constant, then an accepting run of $\mathcal{A}_{\mathcal{G}, (v_1, v_2)} \cap \mathcal{B}'$ can be guessed in logarithmic space, it follows that the upper bound of the data complexity of the evaluation problem of NRRAs is NLOGSPACE.

The PSPACE lower bound is obtained by an easy reduction from the nonemptiness of NRRRA. The NLOGSPACE lower bound of data complexity is from that of RPQs. \square

Theorem 4. *Let $k \geq 1$, $\xi = (x, L, y)$ be a 2RDPQ over the alphabet Σ such that L is given by a NRA or REM containing at most k -registers. Then a 2RRDPQ $\xi' = (x, L', y)$ over the alphabet $\Sigma^\pm \cup \{A_i, A_i^- \mid 1 \leq i \leq k\}$ can be constructed in polynomial time such that for every data graph $\mathcal{G} = (V, E, \eta)$, $\xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn, k})$.*

PROOF. Let (x, L, y) be a 2RDPQ. We first consider the situation that L is given by a NRA $\mathcal{A} = (Q, k, \delta, I, F)$ over the alphabet Σ^\pm .

In the following, we will construct a NRRRA $\mathcal{A}' = (Q', \delta', I', F')$ over $\Sigma^\pm \cup \{A_i, A_i^- \mid 1 \leq i \leq k\}$ so that $\xi' = (x, \mathcal{L}(\mathcal{A}'), y)$ satisfies that $\xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn, k})$.

The intuition of \mathcal{A}' is to simulate the run of \mathcal{A} , by using the following tricks.

Every time a data value d is stored into the i -th register in \mathcal{A} , the sequence $dA_i dA_i^- d$ is read by \mathcal{A}' . Later on, we can refer to the data values stored in the i -th register by using the position terms $pred_{A_i^-}$.

We formally define $\mathcal{A}' = (Q', \delta', I', F')$ as follows.

- $Q' = Q'_w \cup Q'_d$ such that
 - Q'_w is the union of Q_w and $\delta_d \times \{A_i, A_i^- \mid 1 \leq i \leq k\}$,
 - Q'_d is the union of Q_d and $\delta_d \times \{\$i, \#i \mid 1 \leq i \leq k\}$.

- $I' = I, F' = F$.
- $\delta' = \delta'_w \cup \delta'_d$ is defined as follows.
 - $\delta_w \subseteq \delta'_w$.
 - For every transition $(q, c, q', X) \in \delta_d$, let $c' \in C_{rgd}[\Sigma^\pm]$ be obtained from c by replacing every r_j ($1 \leq j \leq k$) with $pred_{A_j^-}$ and r_0 with cur . If $X = \emptyset$, then $(q, c', q') \in \delta'_d$, otherwise, let $X = \{r_{i_1}, \dots, r_{i_\ell}\}$, then δ' includes the following transitions,

$$\begin{aligned}
& q \xrightarrow{c'} ((q, c, q', X), A_{i_1}) \xrightarrow{A_{i_1}} ((q, c, q', X), \$1) \xrightarrow{true} ((q, c, q', X), A_{i_1}^-) \\
& \xrightarrow{A_{i_1}^-} ((q, c, q', X), \#1) \xrightarrow{true} ((q, c, q', X), A_{i_2}) \xrightarrow{A_{i_2}} ((q, c, q', X), \$2) \dots \\
& \xrightarrow{true} ((q, c, q', X), A_{i_\ell}) \xrightarrow{A_{i_\ell}} ((q, c, q', X), \$\ell) \xrightarrow{true} ((q, c, q', X), A_{i_\ell}^-) \\
& \xrightarrow{A_{i_\ell}^-} ((q, c, q', X), \#\ell) \xrightarrow{true} q'
\end{aligned}$$

If L is given by a REM e over the alphabet Σ^\pm , we construct a RREM e' such that $\xi' = (x, L(e'), y)$ over the alphabet $\Sigma^\pm \cup \{A_i, A_i^- \mid 1 \leq i \leq k\}$ satisfies that $\xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn,k})$.

From a REM e , we construct a RREM $tr(e)$ by an induction on the structure of REMs. The nontrivial cases are $e = \downarrow_X e_1$ and $e = e_1[c]$. For $e = \downarrow_X e_1$, suppose $X = \{r_{i_1}, \dots, r_{i_\ell}\}$, then $tr(e) = A_{i_1} A_{i_1}^- \dots A_{i_\ell} A_{i_\ell}^- tr(e_1)$. For $e = e_1[c]$, let $c' \in C_{rgd}$ be obtained from c by replacing r_j with $pred_{A_j^-}$ and r_0 with cur , then $tr(e) = tr(e_1) \cdot [c']$. \square

E. PROOFS IN SECTION 5

Proposition 11. *The evaluation of C2RRDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.*

PROOF. The PSPACE lower bound follows from that of 2RRDPQs. The NLOGSPACE lower bound follows from that of RPQs.

The upper bound:

Let $\xi := Ans(\bar{z}) \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$ be a C2RRDPQ, $\mathcal{G} = (V, E, \eta)$ a data graph, and \bar{v} is a tuple of nodes of the same arity as \bar{z} . Suppose for every $i : 1 \leq i \leq l$, L_i is given by a NRRRA $\mathcal{A}_i = (Q_i, \delta_i, I_i, F_i)$ over the alphabet Σ^\pm .

From the proof of Theorem 3, we know that from each \mathcal{A}_i , an equivalent NRAG $\mathcal{B}_i = (Q'_i, k_i, \delta'_i, I'_i, F'_i)$ can be constructed such that $|Q'_i|$ is polynomial over $|Q_i|$ and exponential over $|\mathcal{T}_{\mathcal{A}_i}|$, and k is polynomial over $|\mathcal{T}_{\mathcal{A}_i}|$.

When restricted to the data paths where all data values are from D , the NRAG \mathcal{B}_i can be seen as a NFA \mathcal{B}'_i over the alphabet $\Sigma^\pm \cup D$ with the state space $Q'_i \times D^{[k_i]}$. It follows that the size of the state space of \mathcal{B}'_i is exponential over the size of \mathcal{A}_i and polynomial over the size of D .

To check whether $\bar{v} \in \xi(\mathcal{G})$, an assignment ν of nodes in V to $\{y_1, \dots, y_{2l}\}$ is first guessed such that $\nu(\bar{z}) = \bar{v}$.

Similarly to the proof of Proposition 10, for every pair $(\nu(y_{2i-1}), \nu(y_{2i}))$, the data graph \mathcal{G} together with $(\nu(y_{2i-1}), \nu(y_{2i}))$ can be seen as a NFA $\mathcal{A}_{\mathcal{G},i}$ over the alphabet $\Sigma^\pm \cup D$ with the initial state $(\nu(y_{2i-1}))_s$ and the final state $(\nu(y_{2i}))_t$.

Then for every $i : 1 \leq i \leq l$, an accepting run of $\mathcal{A}_{\mathcal{G},i} \cap \mathcal{B}'_i$ can be guessed in polynomial space. To check whether $\bar{v} \in \xi(\mathcal{G})$, the accepting runs of the NFAs $\mathcal{A}_{\mathcal{G},i} \cap \mathcal{B}'_i$ can be guessed one by one. From Savitch's theorem, we deduce that the nonemptiness of C2RRDPQs is in PSPACE.

Similarly, if the size of ξ is bounded by a constant, then the assignment ν and the accepting runs of $\mathcal{A}_{\mathcal{G},i} \cap \mathcal{B}'_i$ can be guessed in logarithmic space. Therefore, the evaluation problem of C2RRDPQs has the NLOGSPACE data complexity. \square

Lemma 2. *Suppose $\pi' = v_0 a_1 v_1 \dots a_\ell v_\ell$ is a semipath in \mathcal{G} such that $ur_{v\bar{\pi}}(\pi') = \pi'_0 \# \dots \# \pi'_r$ and $trc(\pi') = p_0 b_1 p_1 \dots b_{\ell+2r} p_{\ell+2r}$ (where $b_1, \dots, b_{\ell+2r} \in \Sigma^\pm \cup \{\#\}$). Then for every $i : 0 \leq i \leq \ell + 2r$, there exists a function $pos_i \in (\mathcal{T}_p[\Sigma_{\xi_1}] \cup \{\perp\})^{\mathcal{T}_A}$ such that for every $t \in \mathcal{T}_A$, $pos_i(t) = \perp$ iff $t_{ur_{v\bar{\pi}}(\pi')}^{adj}[2i] = \perp$; moreover, if $pos_i(t) \neq \perp$ and $t_{ur_{v\bar{\pi}}(\pi')}^{adj}[2i] = 2i'$, then $(pos_i(t))_{\alpha_{\mathcal{G}}}[p_i] = p_{i'}$.*

PROOF. Let $\pi' = v_0 a_1 v_1 \dots a_\ell v_\ell$ be a semipath in \mathcal{G} , $unr_{\bar{\pi}}(\pi') = \pi'_0 \# \dots \# \pi'_r$ such that for every $s : 0 \leq s \leq r$, all the edges on π'_s belonging to π_{j_s} for some $j_s : 1 \leq j_s \leq l_1$, and $trc(\pi') = p_0 b_1 p_1 \dots b_{\ell+2r} p_{\ell+2r}$.

We prove the lemma by an induction on the structure of position terms.

Induction base: For every $i : 0 \leq i \leq \ell + 2r$, $pos_i(cur) = cur$.

Induction step:

Let us first consider the case $t = \text{suc}(t_1)$.

Let $i : 0 \leq i \leq \ell + 2r$.

If $t_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = \perp$, then let $\text{pos}_i(t) = \perp$. Otherwise, $t_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = 2i'$ for some i' . From $t = \text{suc}(t_1)$, it follows that $(t_1)_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = 2i''$ for some i'' such that $2i' = 2i'' + 2$ or $2i' = 2i'' + 4$.

According to the induction hypothesis, there exists $\text{pos}_i(t_1) \in \mathcal{T}_p[\Sigma_{\xi_1}]$ such that $(\text{pos}_i(t_1))_{\alpha_{\mathcal{G}}}[p_i] = p_{i''}$.

- If $2i' = 2i'' + 2$, then let $\text{pos}_i(t) = \text{pos}_i(\text{suc}(t_1)) = \text{suc}(\text{pos}_i(t_1))$ if $p_{i''} < p_{i'}$, otherwise, let $\text{pos}_i(t) = \text{pos}_i(\text{suc}(t_1)) = \text{pred}(\text{pos}_i(t_1))$.
- If $2i' = 2i'' + 4$, then there are $j_1, j_2 : 1 \leq j_1, j_2 \leq l_1$ such that one of the following conditions holds,
 1. $p_{i''}$ is the position immediately before $\$_{2j_1}$, $p_{i'}$ is the third position before $\$_{2j_2}$ in $\alpha_{\mathcal{G}}$, and $j_1 \neq j_2$,
 2. $p_{i''}$ is the position immediately before $\$_{2j_1}$, $p_{i'}$ is the third position after $\$_{2j_2-1}$,
 3. $p_{i''}$ is the position immediately after $\$_{2j_1-1}$, $p_{i'}$ is the third position before $\$_{2j_2}$,
 4. $p_{i''}$ is the position immediately after $\$_{2j_1-1}$, $p_{i'}$ is the third position after $\$_{2j_2-1}$, and $j_1 \neq j_2$.

We illustrate the argument by considering the second situation above. The arguments for the other three situations are similar.

- if $j_1 < j_2$, then let $\text{pos}_i(\text{suc}(t_1)) = \text{suc}(\text{suc}_{\$_{2j_2-1}}(\text{pos}_i(t_1)))$,
- if $j_1 > j_2$, then let $\text{pos}_i(\text{suc}(t_1)) = \text{suc}^2(\text{pred}_{\$_{2j_2-1}}(\text{pos}_i(t_1)))$.

The case $t = \text{pred}(t_1)$ can be discussed similarly as $t = \text{suc}(t_1)$.

Now consider the case $t = \text{suc}_A(t_1)$.

Let $i : 0 \leq i \leq \ell + 2r$.

If $t_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = \perp$, let $\text{pos}_i(t) = \perp$. Otherwise, let $t_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = 2i'$. From $t = \text{suc}_A(t_1)$, we know that $(t_1)_{\text{urv}\bar{\pi}}^{\text{adj}}(\pi')[2i] = 2i''$ for some i'' such that $2i'' < 2i'$.

From the induction hypothesis, $(\text{pos}_i(t_1))_{\alpha_{\mathcal{G}}}[p_i] = p_{i''}$.

If there are no $\#$ symbols in the subpath of $\text{urv}\bar{\pi}(\pi')$ from the position $2i''$ to $2i'$, then the position $2i''$ and $2i'$ both belong to π'_s for some $s : 0 \leq s \leq r$. It follows that $p_{i''}$ and $p_{i'}$ are two positions between $\$_{2j_s-1}$ and $\$_{2j_s}$ in $\alpha_{\mathcal{G}}$. Define $\text{pos}_i(t)$ as follows.

- If $p_{i'} < p_{i''}$, let $\text{pos}_i(\text{suc}_A(t_1)) = \text{pred}_{A \times \{j_s\}}(\text{pos}_i(t_1))$.
- If $p_{i''} < p_{i'}$, let $\text{pos}_i(\text{suc}_A(t_1)) = \text{suc}_{A \times \{j_s\}}(\text{pos}_i(t_1))$.

Otherwise (that is, there are $\#$ symbols from $2i''$ to $2i'$), let $2i'''$ be the position before the position $2i'$ on $\text{urv}\bar{\pi}(\pi')$ such that $2i'''$ is a position immediately after $\#$ and $2i'''$ is the last position before $2i'$ satisfying this property. Let $s : 0 \leq s \leq r$ such that $2i'''$ and $2i'$ are two positions belonging to π'_s . Then $p_{2i'''}$ is the position immediately after $\$_{2j_s-1}$ or the position immediately before $\$_{2j_s}$ in $\alpha_{\mathcal{G}}$. We illustrate the argument by considering the situation that $p_{2i'''}$ is the position immediately after $\$_{2j_s-1}$. The discussion for the latter situation is similar. Define $\text{pos}_i(t)$ as follows.

- If $p_{i'} < p_{i''}$, let $\text{pos}_i(\text{suc}_A(t_1)) = \text{suc}_{A \times \{j_s\}}(\text{pred}_{\$_{2j_s-1}}(\text{pos}_i(t_1)))$.
- If $p_{i''} < p_{i'}$, let $\text{pos}_i(\text{suc}_A(t_1)) = \text{suc}_{A \times \{j_s\}}(\text{suc}_{\$_{2j_s-1}}(\text{pos}_i(t_1)))$.

The case $t = \text{pred}_A(t_1)$ can be discussed similarly to $t = \text{suc}_A(t_1)$.

In summary, for every $t = \text{op}(t_1) \in \mathcal{T}_p[\Sigma^\pm]$ such that $\text{pos}_i(t) \neq \perp$ (where $\text{op} = \text{suc}, \text{pred}, \text{suc}_A, \text{pred}_A$), there is $t_{\text{op}} \in \mathcal{T}_p[\Sigma_{\xi_1}]$ such that $\text{pos}_i(t) = t_{\text{op}}[\text{cur} \setminus \text{pos}_i(t_1)]$. \square

Theorem 6. Let \mathcal{G} be a ν -canonical data graph for ξ_1 , ξ be a 2RRDPQ. Then a 2NRRRA \mathcal{A}_ξ can be constructed from ξ and ξ_1 such that $\xi(\mathcal{G})$ is nonempty iff \mathcal{A}_ξ accepts $\vdash \alpha_{\mathcal{G}} \dashv$.

PROOF. Let π' be a path in \mathcal{G} , the $\bar{\pi}$ -unraveling of π be $\pi'_0 \dots \pi'_r$, where for every $s : 0 \leq s \leq r$, all the edges on $\pi'_s = v_{i_s} a_{i_s+1} v_{i_s+1} \dots v_{i_{s+1}}$ belong to π_{j_s} .

Our goal is to construct a 2NRRRA \mathcal{B} to simulate the runs of \mathcal{A}' over $\eta(\text{urv}\bar{\pi}(\pi'))$.

Similarly to the transformation from NRRAs to NRAGs in Theorem 3, the 2NRRRA \mathcal{B} goes through $\text{trc}(\pi')$ in $\alpha_{\mathcal{G}}$ and guesses the profile of the current position of $\eta(\text{urv}\bar{\pi}(\pi'))$, in order to simulate \mathcal{A}' over $\eta(\text{urv}\bar{\pi}(\pi'))$. The difference is that instead of storing and guessing the data values, \mathcal{B} records and guesses a position term from $\mathcal{T}_p[\Sigma_{\xi_1}]$ (interpreted over $\alpha_{\mathcal{G}}$) for each position term occurring in the profile of the current position in $\eta(\text{urv}\bar{\pi}(\pi'))$. The intricacy of the construction is how to guarantee the consistency of the guessed position terms $\mathcal{T}_p[\Sigma_{\xi_1}]$ and how to update them during the simulation.

A *locating profile* loc of \mathcal{A}' over $\alpha_{\mathcal{G}}$, is defined as a pair $((S, \chi, \sim), pos)$, where $(S, \chi, \sim) \in Prof_{\mathcal{A}}$ (cf. proof of Theorem 3),

$$\chi = \frac{(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \dots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})}{(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \dots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})}$$

and $pos : (\mathcal{T}_p[\Sigma_{\xi_1}] \cup \{\perp\})^{\mathcal{T}_{\mathcal{A}}}$ such that

- $pos(t) = \perp$ for every $t \in \mathcal{T}_{\mathcal{A}} \setminus S$,
- $pos(cur) = cur$,
- for every $t, t' \in S$ such that $t \leq t'$, we have $pos(t) \leq pos(t')$,
- for every $t_1, t_2 \in S$ such that there is $j : -m_1 \leq j \leq m_2$ satisfying that $t_1, t_2 \in T_j$, if $op(t_1), op(t_2) \in S$ for $op \in \{suc, pred, suc_A, pred_A \mid A \subseteq \Sigma^{\pm}\}$, then there is $t_{op} \in \mathcal{T}_p[\Sigma_{\xi_1}]$ of the form as those in the proof of Lemma 2 (e.g. $t_{suc} = suc(suc_{\S_{2j_2-1}}(cur))$) such that $pos(op(t_j)) = t_{op}[cur \setminus pos(t_j)]$ for $j = 1, 2$.

Let Σ_{loc} denote the set of locating profiles.

Similar to the construction of NRAGs from NRRAs, we define two successor relations between locating profiles.

Let $((S_1, \chi_1, \sim_1), pos_1), ((S_2, \chi_2, \sim_2), pos_2) \in \Sigma_{loc}$, $a \in \Sigma^{\pm}$, $j : 1 \leq j \leq l_1$, $1 \leq k_1, k_2 \leq 2l_1$, and $dir \in \{+1, -1\}$.

In the following, we will define two relations $((S_1, \chi_1, \sim_1), pos_1) \xrightarrow{((a, j), dir)} ((S_2, \chi_2, \sim_2), pos_2)$ and $((S_1, \chi_1, \sim_1), pos_1) \xrightarrow{(a, \S_{k_1}, \S_{k_2})} ((S_2, \chi_2, \sim_2), pos_2)$. The latter relation corresponds to the situation that the run of \mathcal{A}' is jumping over $\#$ on $urv_{\pi}(\pi')$, and the former relation corresponds to the situation that the run of \mathcal{A}' is not.

At first, $((S_1, \chi_1, \sim_1), pos_1) \xrightarrow{((a, j), dir)} ((S_2, \chi_2, \sim_2), pos_2)$ if the following conditions hold.

- $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$.
- If $suc(cur) \in S_1$, then $pos_1(suc(cur)) = suc(cur)$ if $dir = +1$, and $pos_1(suc(cur)) = pred(cur)$ otherwise.
- If $a \in A$ and $suc_A(cur) \in S_1$, then $pos_1(suc_A(cur)) = suc_{A \times \{j\}}(cur)$ if $dir = +1$, and $pos_1(suc_A(cur)) = pred_{A \times \{j\}}(cur)$ otherwise.
- For every $t \in S_1$ such that $suc(cur) \leq t$, if $dir = +1$, then $pos_2(t[suc(cur) \setminus cur]) = pos_1(t)[suc(cur) \setminus cur]$, otherwise, $pos_2(t[suc(cur) \setminus cur]) = pos_1(t)[pred(cur) \setminus cur]$.
- For every $t \in S_1$ such that $suc_A(cur) \leq t$ and $a \in A$, if $dir = +1$, then

$$pos_2(t[suc_A(cur) \setminus cur]) = pos_1(t)[suc_{A \times \{j\}}(cur) \setminus cur],$$

otherwise, $pos_2(t[suc_A(cur) \setminus cur]) = pos_1(t)[pred_{A \times \{j\}}(cur) \setminus cur]$.

- For every $t \in S_2$ such that $pred(cur) \leq t$, $pos_2(t) = pos_1(t[pred(cur) \setminus cur])$.
- For every $t \in S_2$ such that $pred_A(cur) \leq t$ and $a \in A$, $pos_2(t) = pos_1(t[pred_A(cur) \setminus cur])$.
- For every $t \in S_1$ such that $suc_A(cur) \leq t$ and $a \notin A$, $pos_2(t) = pos_1(t)$.
- For every $t \in S_1$ such that $pred_A(cur) \leq t$ and $a \notin A$, $pos_2(t) = pos_1(t)$.

In the following, we will define $((S_1, \chi_1, \sim_1), pos_1) \xrightarrow{(a, \S_{k_1}, \S_{k_2})} ((S_2, \chi_2, \sim_2), pos_2)$ for k_1, k_2 satisfying that there are $k'_1, k'_2 : 1 \leq k'_1, k'_2 \leq l_1$ such that one of the following conditions hold.

1. $k_1 = 2k'_1$, $k_2 = 2k'_2$, and $k'_1 \neq k'_2$,
2. or $k_1 = 2k'_1$, $k_2 = 2k'_2 - 1$,
3. or $k_1 = 2k'_1 - 1$, $k_2 = 2k'_2$,
4. $k_1 = 2k'_1 - 1$, $k_2 = 2k'_2 - 1$, and $k'_1 \neq k'_2$.

We will illustrate the definition for the first case above, the other three cases can be discussed in the same way.

Suppose $k_1 = 2k'_1$, $k_2 = 2k'_2$, and $k'_1 \neq k'_2$ for some k'_1, k'_2 . Then $((S_1, \chi_1, \sim_1), pos_1) \xrightarrow{(a, \S_{2k'_1}, \S_{2k'_2})} ((S_2, \chi_2, \sim_2), pos_2)$ if the following conditions hold.

- $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$.
- If $suc(cur) \in S_1$, then $pos_1(suc(cur)) = pred^2(suc_{\S_{k_2}}(cur))$ if $k'_1 < k'_2$, and $pos_1(suc(cur)) = pred(pred_{\S_{k_2}}(cur))$ otherwise.

- If $a \in A$ and $\text{suc}_A(\text{cur}) \in S_1$, then $\text{pos}_1(\text{suc}_A(\text{cur})) = \text{pred}_{A \times \{k'_2\}}(\text{suc}_{\$_{k_2}}(\text{cur}))$ if $k'_1 < k'_2$, and $\text{pos}_1(\text{suc}_A(\text{cur})) = \text{pred}_{A \times [k'_2]}(\text{pred}_{\$_{k_2}}(\text{cur}))$ otherwise.
- For every $t \in S_1$ such that $\text{suc}(\text{cur}) \leq t$, if $k'_1 < k'_2$, then

$$\text{pos}_2(t[\text{suc}(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}^2(\text{suc}_{\$_{k_2}}(\text{cur})) \setminus \text{cur}] ,$$

otherwise,

$$\text{pos}_2(t[\text{suc}(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}(\text{pred}_{\$_{k_2}}(\text{cur})) \setminus \text{cur}] .$$

- For every $t \in S_1$ such that $\text{suc}_A(\text{cur}) \leq t$ and $a \in A$, if $k'_1 < k'_2$, then

$$\text{pos}_2(t[\text{suc}_A(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}_{A \times \{k'_2\}}(\text{suc}_{\$_{k_2}}(\text{cur})) \setminus \text{cur}] ,$$

otherwise,

$$\text{pos}_2(t[\text{suc}_A(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}_{A \times [k'_2]}(\text{pred}_{\$_{k_2}}(\text{cur})) \setminus \text{cur}] .$$

- For every $t \in S_2$ such that $\text{pred}(\text{cur}) \leq t$, $\text{pos}_2(t) = \text{pos}_1(t[\text{pred}(\text{cur}) \setminus \text{cur}])$.
- For every $t \in S_2$ such that $\text{pred}_A(\text{cur}) \leq t$ and $a \in A$, $\text{pos}_2(t) = \text{pos}_1(t[\text{pred}_A(\text{cur}) \setminus \text{cur}])$.
- For every $t \in S_1$ such that $\text{suc}_A(\text{cur}) \leq t$ and $a \notin A$, $\text{pos}_2(t) = \text{pos}_1(t)$.
- For every $t \in S_1$ such that $\text{pred}_A(\text{cur}) \leq t$ and $a \notin A$, $\text{pos}_2(t) = \text{pos}_1(t)$.

We are ready to construct the 2NRRRA \mathcal{B} .

Suppose $\pi' = v_0 a_1 v_1 \dots a_\ell v_\ell$ is a semipath in $\alpha_{\mathcal{G}}$, $\text{urv}_{\pi}(\pi') = \pi'_0 \# \dots \# \pi'_r$, for every $s : 0 \leq s \leq r$, all the edges on π'_s belong to π_{j_s} ($1 \leq j_s \leq l_1$), and $\text{trc}(\pi') = d_0 a'_1 p_1 \dots a'_{\ell+2r} p_{\ell+2r}$ (where for every $j : 1 \leq j \leq \ell+2r$, $a'_j \in \Sigma^\pm \cup \{\#\}$). Then \mathcal{B} does the following.

- In each position p_i ($0 \leq i \leq \ell+2r$) of $\alpha_{\mathcal{G}}$, \mathcal{B} guesses a locating profile $\text{loc}_i = ((S_i, \chi_i, \sim_i), \text{pos}_i) \in \Sigma_{\text{loc}}$ with

$$\chi_i = \begin{pmatrix} (b_{i,-m_{i,1}}, T_{i,-m_{i,1}}, b'_{i,-m_{i,1}}, s_{i,-m_{i,1}}) \dots (b_{i,-1}, T_{i,-1}, b'_{i,-1}, s_{i,-1}) \\ (b_{i,0}, T_{i,0}, b'_{i,0}, s_{i,0}) (b_{i,1}, T_{i,1}, b'_{i,1}, s_{i,1}) \dots (b_{i,m_{i,2}}, T_{i,m_{i,2}}, b'_{i,m_{i,2}}, s_{i,m_{i,2}}) \end{pmatrix} .$$

In addition,

- if $i = 0$, then $\text{loc}_0 = (S_0, \chi_0, \sim_0)$ is an initial locating profile, that is, for every $t \in \mathcal{T}_A$ such that $\text{pred}(\text{cur}) \leq t$ or $\text{pred}_A(\text{cur}) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_0$,
- if $i = \ell+2r$, then $\text{loc}_{\ell+2r} = (S_{\ell+2r}, \chi_{\ell+2r}, \sim_{\ell+2r})$ is a final profile, that is, for every $t \in \mathcal{T}_A$ such that $\text{suc}(\text{cur}) \leq t$ or $\text{suc}_A(\text{cur}) \leq t$ for some $A \subseteq \Sigma^\pm$, $t \notin S_{\ell+2r}$.
- Over each pair of positions p_i and p_{i+1} (where $0 \leq i < \ell+2r$) of $\alpha_{\mathcal{G}}$,
 - if $i_s + 2s \leq 2i < i_{s+1} + 2s$ for some $s : 0 \leq s \leq r$, then \mathcal{B} checks that $\text{loc}_i \xrightarrow{(a'_{i+1}, j_s), \text{dir}} \text{loc}_{i+1}$, where $\text{dir} = +1$ if $p_{i+1} = p_i + 2$, and $\text{dir} = -1$ otherwise,
 - if $2i = i_s + 2(s-1)$ for some $s : 1 \leq s \leq r$ (that is, $2i$ is the position immediately before $\#$ in $\text{urv}_{\pi}(\pi')$), then \mathcal{B} jumps from p_i to p_{i+1} , then to p_{i+2} , and checks that $\text{loc}_i \xrightarrow{(a'_{i+2}, \$_{k_s}, \$_{k'})} \text{loc}_{i+2}$, where
 - $k = 2j_{s-1}$ if $p_{\alpha_{\mathcal{G}}}(\pi_{j_{s-1}}, v_{i_s})$ is the position immediately before $\$_{2j_{s-1}}$, and $k = 2j_{s-1} - 1$ if $p_{\alpha_{\mathcal{G}}}(\pi_{j_{s-1}}, v_{i_s})$ is the position immediately after $\$_{2j_{s-1}-1}$,
 - $k' = 2j_s$ if $p_{\alpha_{\mathcal{G}}}(\pi_{j_s}, v_{i_s})$ is the position immediately before $\$_{2j_s}$, and $k' = 2j_s - 1$ if $p_{\alpha_{\mathcal{G}}}(\pi_{j_s}, v_{i_s})$ is the position immediately after $\$_{2j_s-1}$.
- At the same time, \mathcal{B} simulates the run of \mathcal{A}' over $\eta(\text{urc}_{\pi}(\pi'))$ as follows.
 - If \mathcal{A}' makes a transition (q, a_i, q') over a_i , then \mathcal{B} checks that $b'_{i-1,0} = a_i$ and changes the state from q to q' .
 - If \mathcal{A}' makes a transition (q, c, q') in the position $2i$ of $\eta(\text{urc}_{\pi}(\pi'))$, then \mathcal{B} checks that (S_i, χ_i, \sim_i) satisfies c , verifies that the data value in the current position is equal to the data value in the position represented by $\text{pos}_i(t)$ for each $t \in S_i$ such that $\text{cur} \sim_i t$, and changes the state from q to q' .
 - \mathcal{B} accepts if \mathcal{A}' accepts and a final profile is reached.

From the above construction, we know that in its states, \mathcal{B} should record the states of \mathcal{A}' and the guessed locating profiles. Because both the number of profiles and the number of functions pos in locating profiles are exponential over $|\mathcal{T}_A|$, it follows that the number of states of \mathcal{B} is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. \square